# SESSION

# CYBER-PHYSICAL SYSTEMS AND APPLICATIONS

## Chair(s)

**TBA**

ISBN: 1-60132-433-2, CSREA Press ©

ISBN: 1-60132-433-2, CSREA Press ©

# Detection of Human Limits in Hazardous Environments: A Human-Centric Cyber-Physical System

**Constantin Scheuermann[1], Raman Tandon[2], Bernd Bruegge[1] and Stephan Verclas[3]**
[1]Department of Computer Science, Technical University Munich, Munich, Germany
[2]Wehrwissenschaftliches Institut fuer Werk- und Betriebsstoffe, Erding, Germany
[3]T-Systems International GmbH, Munich, Germany

**Abstract**— *Harsh and hazardous working conditions can push humans to their limits when completing their tasks. Smart Textiles, wearables and 3D printing technology enable us to form Human-Centric Cyber-Physical Systems (HCCPSs) that put the human in the center of the system design. HCCPSs aim to increase operational safety by monitoring for example health parameters and providing support to humans. Wrong or even not present human risk management, the presence of physiological stress and human information overload can have dramatic consequences in harsh and hazardous working environments. Thus, unclear User Interface design, unpractical monitoring techniques as well as complicated communication channels can impede human support. We brought together researchers from different domains and collaborated with the startup (ambiotex) to form the vertical prototype Cooling Health Safety Textile (CHEST). CHEST is a browser based remote health monitoring system using easy applicable smart textiles and a lean communication channel together with an app based visualization software. We offer a lean User Interface to minimize information overload and physiological stress. Currently, we are working on the integration of a smart cooling system integrated into the worker's textile.*

**Keywords:** Physiological Stress, Health Monitoring, Human-Centric Cyber-Physical Systems, MQTT, Smart Textiles, Wearables

## 1. Introduction

Technical improvements over the last 70 years are enabling researchers and industries to build Human-Centric Cyber-Physical Systems (HCCPSs). Especially the miniaturization of sensors, actuators and computer chips made it possible to generate "wearable" IT-devices. HCCPSs are bridging the cyber and IT environment and the physical environment that includes human beings. HCCPSs put the human into the center of the system design. Combining technologies such as tiny, energy-efficient mobile devices with applicable sensors and actuators is a requirement to integrate smart systems seamless into daily objects. The use of smart textiles in combination with wearables allows us to focus on humans and reconceive as well as extend existing concepts and processes. HCCPS consider humans as part of the system. The human condition as well as human behavior serve as system input. Such input can be error prone. Feedback and communication channels are essential to allow quasi real-time system state monitoring, especially for HCCPSs.

Within HCCPSs erroneous behavior and its results are a major issue to address. Such erroneous behavior can lead to operational safety risks. Fig. 1 shows the influence factors that can increase or decrease the probability of errors to happen. Erroneous behavior can be found in different layers within a HCCPS, either in the human environment or in the cyber and IT environment (Software, Network and Hardware). Within HCCPSs the human is the center of the system design and the cyber and IT environment needs to be human orientated in order to reduce human based errors. As one might know from every day life, the more things we do in parallel the more stress we feel. The same applies for information overload while using IT systems. The caused physiological stress can create human induced errors. Especially, in harsh working environments, stress, great responsibility, dependencies and the risk of lives are prominent challenges for humans. Errors in such environments need to be reduced or even avoided using a system that tries to reduce or eliminate physiological stress factors.

According to James Reason, human errors can be categorized as *Mistakes*, *Lapses* and *Slips*. *Mistakes* refers to an incorrect plan while identifying a goal and selecting the tools to achieve the goal. *Lapses* occur in case the chosen plan, to achieve a certain goal, is postponed, and needs to be stored. In case the plan needs to be memorized and thus a longer time period passes, errors can occur. *Slips* are errors that appear during an action [1]. Sanders and McCormick define human error as "an inappropriate or undesirable human decision or behavior that reduces, or has the potential of reducing effectiveness, safety, or system performance" [3]. In literature, human errors are also referred to as human-made faults caused by human actions or the absence of human action. They have phenomenological causes as well as physical hardware faults. [4].

Physical Hardware faults are natural faults caused by natural phenomena without human participation. Software faults such as software bugs are widely addressed in the research field of software engineering. Concepts as continuous integration or test driven development try to keep such errors

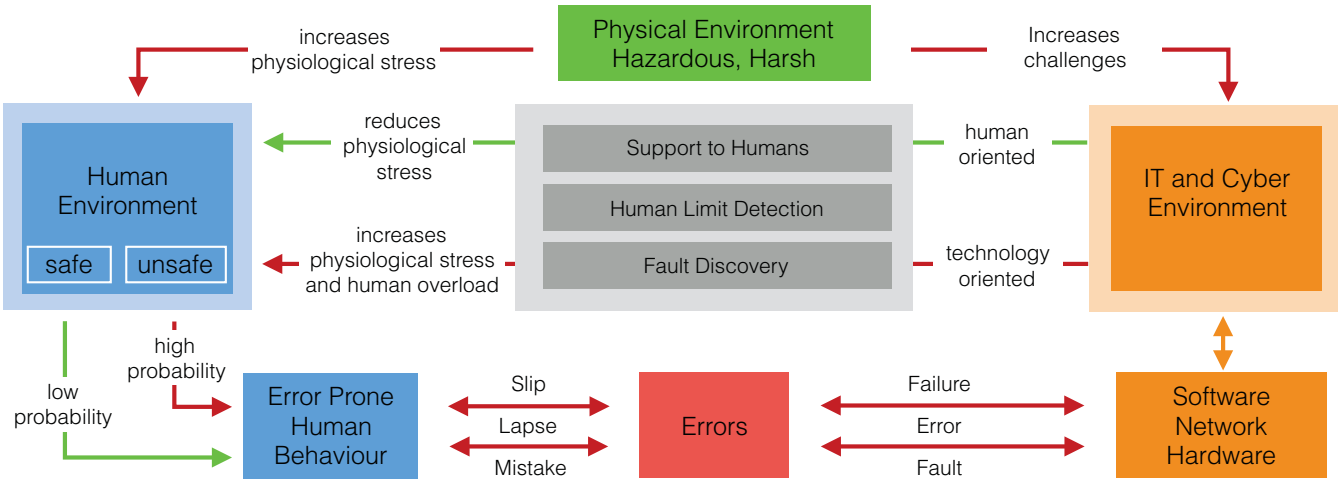



Fig. 1: **Focus of HCCPSs:** Human-Centric Cyber-Physical Systems have to deal with high complexity due to the included subsystems. Human behavior is error prone as humans are doing mistakes, lapses and slips [1]. The Software, Network and Hardware subsystems can contain faults that result in errors and in the worst case cause failure. As the human is the center of the system, information overload and physiological stress can be the result. HCCPSs try to decrease these factors by offering Support to Humans, Detection of Human Limits and Fault Discovery. Human oriented design can decrease physiological stress while technology oriented design can increase information overload and physiological stress. The Illustration is based on [1] [2] and extended.

(bugs) at a minimum. Radio signal jamming, signal hijacking and physical phenomena such as fading and pathloss can for example cause network faults. [4] [5]

As shown in Fig. 1 the error complexity covers 3 layers: Human environment, Physical Environment and the IT and Cyber environment. We state that HCCPSs cannot be error free and are highly error prone due to the complexity of potential error sources. As humans are the center of the system design, Support to Humans, Human Limit Detection and Fault Discovery are aspects we want to address. We suggest deploying real-time human observers, which can monitor the system concerning all three error layers.

Humans are still an important part of professional work routines and processes in industries. In professional hazardous environments they work in tight cooperation with robots. Yet for scenarios where robots cannot access or the environment is just not suitable for a robot, humans have to operate. In this study we focus on risk minimization for the workers wearing CBRN-protective suits through permanent monitoring of the vital signs utilizing a smart textile. Since these CBRN-suits are totally impermeable the worker is exposed to dramatic heat stress and thus can easily collapse. Therefore, medical staff monitors via wireless live data transmission the vital signs of the worker wearing a CBRN-suit. The Ebola virus disease or the Fukushima catastrophe, are typical scenarios where humans have to operate in harsh environments, wearing CBRN-suits. Currently, real-time remote health monitoring, including a feedback channel between the worker, a medcial advisor (MC) and a commander, during such missions is difficult to achieve[1]. Current monitoring systems allow only an analysis of data after the mission has ended and real-time monitoring is yet in its infancy. Additionally, we investigate on a Human Limit Detection system offering real-time observer monitoring capabilities.

Self monitoring capabilities as well as health stress level dependent cooling capabilities should contribute to a better wearing comfort of CBRN suits, decrease physiological stress of the worker and result in longer working times than now. Currently, humans wearing such suits are only allowed to operate 30 minutes[2]. We propose smart cooling systems considering the health condition of the humans wearing CBNR suits. The paper is structured as follows: Section 2 describes the related work. Section 3 explains the problem we want to address followed by Section 4 which explains the Visionary Scenario. In Section 5 we describe the system design and then conclude with Section 6 and 7.

## 2. Related Work

We state that human health monitoring is the prerequisite to allow operational safety in professional harsh working environments. With the invention of the stethoscope by Rene Laennec in 1816, a first step towards health monitoring was done. Further inventions improved the first stethoscope such

[1]Personal Discussions with Dr. Raman Tandon and his collegues

[2]DGUV (Deutsche gesetzliche Unfallversicherung) http://publikationen.dguv.de/dguv/pdf/10002/r-189.pdf and http://publikationen.dguv.de/dguv/pdf/10002/r-190.pdf

as the ECG with the contribution of Willem Einthoven. With the invention of the Holter-Monitor, an electrocardiography device that can monitor the activities of the cardiovascular system, the first continuous health monitoring was possible. In sports, heart rate monitors became popular since the 1990s [6]. Nowadays, a variety of devices are commercially available and monitoring systems are even integrated into smart textiles[3].

During the last decades, health-monitoring is an emerging field in research. Research efforts cover structural [7] as well as human health monitoring applications. An extensive overview of existing applications especially concerning patient health monitoring can be found at [8].

Generally, health monitoring systems can be wire-based or wireless. Wire-based systems have major disadvantages such as being tethered to one location, they typically support a short observation period and cannot be placed to real environments [9]. The major advantage is that wire-based systems profit from a reliable data connection. Still considering scenarios where the patient is mobile, cable connections can be a major impediment. To improve user's comfort the usage of conductive yarns, woven into a textile form conductive textiles: so called 'Smart Textiles'. Wireless systems have to face typical challenges such as power consumption, and limited energy resources, security, fading and pathloss, signal jamming as well as limited bandwidth. The major advantage is the location independent operation capability.

Concerning human health monitoring several terms are currently present in research. *(Wireless) Body (Area/Sensor) Networks*, *Wireless Medical Telemetry Services*, *Wireless physiological measurement systems*, *Wearable Wireless Health-Monitoring System (WWHMS)*.

*(Wireless) Body (Area/Sensor) Networks* have their origin in the medical and health domain. The terms first appeared in literature in the early 21st century. Such systems focus on the aging population to address common chronic age related diseases. Typical representatives of these diseases are congestive heart failure, chronic obstructive pulmonary disease, arthritis, osteoporosis, and dementia [9] [10]. Typical challenges are scalability, security and privacy [10]. To realize *Wireless Medical Telemetry Services* the Federal Communication Commission (FCC) allocated the WMTS spectrum that is used for remote patient health monitoring applications. Wireless medical telemetry systems include devices to measure patients' vital signs and other important health parameters (e.g., pulse and respiration rates).[4]

*Wireless physiological measurement systems* (WPMS) make use of wireless transmission capabilities. WPMSs allow real-time physiological data measurements originated from wearable/implantable medical sensors. Data is collected and transmitted to a central processing unit. The primary purpose of the WPMS is to improve the quality and efficiency of healthcare (GE Healthcare 2007).

*Wearable Wireless Health-Monitoring Systems* (WWHMS) track the individual's activities continuously, transmitting the relevant information back to medical professionals. WWHMSs automatically alert health-care personnel when an emergency occurs. The major goal is to prevent delays of emergency medical services in the event of accidents. Delays can often result in serious physical and psychological consequences, permanent disabilities, or even fatalities. [10]

Several standards exist that are commonly used for the above-mentioned health monitoring systems: IEEE 802.15.6, ZigBee, Bluetooth, Bluetooth low energy, Infrared, UWB, Sensium and ANT. The standards mainly focus on the constrain resources such systems have to cope with: power efficiency, energy scavenging or power harvesting, robust communication and high throughput. Other challenges are the form factor, operational safety and security, clinical effectiveness of sensor technology, artificial intelligence and decision support as well as security [10].

As we are combining real-time human condition monitoring, remote monitoring and smart textile cooling concepts, some related work in these research areas is provided. Projects such as the LifeGuard [11] system can monitor two standard electrocardiogram leads, respiration rate via impedance plethysmography, heart rate, hemoglobin oxygen saturation, ambient or body temperature, three axes of acceleration, and blood pressure. Real-time health monitoring in harsh environments is essential for the worker in order to ensure a safe working environment.

Autonomous water and gas based cooling systems already

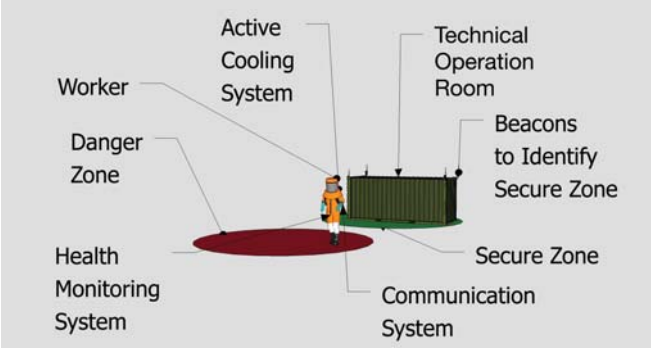



Fig. 2: **Visionary Scenario:** Illustrating the future scenario divided into the secure and danger zone. The worker is equipped with an active cooling system, a communication system and a health monitoring system. The secure zone provides the independent communication infrastructure and an identification technology to mark the secure zone.

[3]Ambiotex Shirt, Zephyr Bioharness, Vivonetics Life Shirt, Hexoskin, Athos Muscleshirt, Health-Lab

[4]https://www.fcc.gov/general/wireless-medical-telemetry-service-wmts Allocated frequency band by FCC for Wireless Medical Telemetry Services (WMTS) in the 608–614, 1395–1400, and 1427–1432 MHz range.

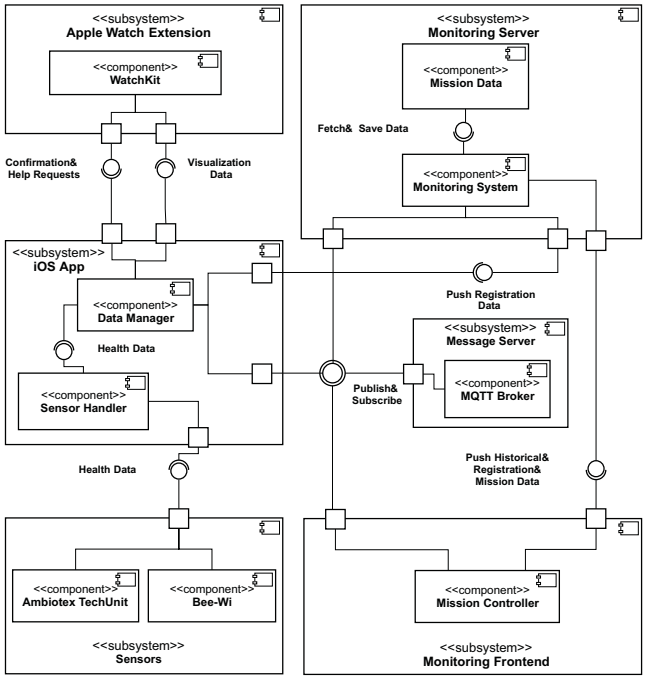



Fig. 3: **Component Diagram:** Showing the provided and used services of each subsystem.

exist such as the combo cool system[5]. Also heated undergarment[6], as well as socks and insoles and gloves[7] are already commercially available.

Latest advances within 3D printing technologies and 3D weaving technologies[8] allow us to integrate and print cooling structures directly as textiles-based structures.

The combination of remote health monitoring systems, smart textiles and intelligent cooling systems enables us to develop a Human-Centric Cyber-Physical System, putting the human and his condition into the center of the system design. We want to transfer the knowledge from WWHMS to professional applications in harsh, hazardous environments. In our definition harsh environments require special kind of heavy-duty equipment such as CBRN-suits that are totally sealed off from the environment. In this case, cooling is of major concern as workers in such suits face the danger of collapsing and suffer from severe thermal stress.

## 3. Problem

Humans perform tasks in several areas. Tasks cover exhausting physical and mental work. Especially during physically exhausting tasks, humans can benefit from information about their health condition. For a professional football player the information about his heart rate can provide a major benefit, as the player should meet a certain training target heart rate. For a professional fire fighter, knowledge about the health condition can improve operational safety during a mission. A collapse, as the most drastic consequence while performing a task in hash environments, needs to be avoided. It can not only influence the operational safety of the fire fighter, but moreover it can influence the entire mission outcome including the involved team members.

To monitor the health condition, several applications are already available for consumers as well as for professionals. For professional athletes such applications help to monitor, control and optimize their performance and training schedule. For professionals operating in harsh environments such monitoring systems can be used to realize remote health monitoring in combination with smart cooling capabilities. Especially in hazardous environments, where CBRN-suits are a requirement, the danger of collapse resulting in a human error is a proper risk.

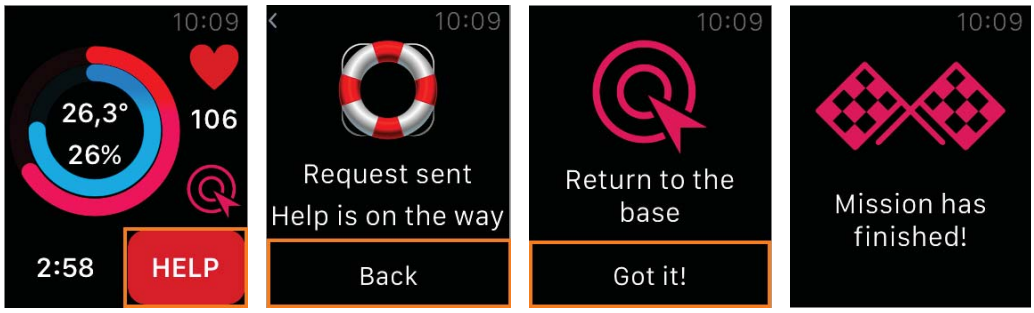
If we are imagining a worker operating in an environment that is contaminated, the isolation of his working suit is a requirement to guarantee operational safety. As the suit is totally isolated, air exchange with the environment is up to impossible. Due to the lack of air exchange several consequences appear. In addition to high environmental temperatures the muscles of the worker produce heat during the operation. This heat increases the temperature within the suit to a level where the worker starts to perspirate. In case we assume a normal workload of 120 Watt a worker operating in a CBRN-Suit evaporates/perspirates up to 0.5 l of sweat in one hour that currently accumulates inside the suit. The sweat accumulates at the tip of the limbs and parts such as the back, head and the chest. The humidity causes swollen, soft finger and toe tips that can cause injuries or can impede a proper task completion. As a result of the thermal stress the heart-rate and blood pressure increase making a collapse more likely. An experiment conducted by Glitz et. al. showed that taking a break would stop muscle heat production but has no significant effect on the thermoregulation of the wearer, since the CBRN-suit is impermeable. For the experiment 10 participants were wearing protective overalls and completed a 130 minutes work-rest schedule in non-ventilated (non-vent.) and ventilated (vent.) thermal insulating protective clothing. It was shown while wearing the ventilated suit the heart rate during the stops almost dropped to normal (90 b/min) whereas in the non-vent. situation the participants' heartrate steadily increased up to 150 b/min and only dropped to 135 b/min during the last break [12]. For professionals, operating in harsh environments, physical overstrain can result in the risk of life for each member of the task force and even for the entire team. Currently, such workers are not allowed to operate more than 30 minutes. This period of time is

[5] http://wolfengineering.de/personen-kuehlung

[6] http://voltheat.com/, Alpenheat, WarmX, Therm-ic, Reusch, Interactive Wear.

[7] Alpenheat, WarmX, Therm-ic, Reusch, Interactive Wear

[8] 3D Weaver http://www.sosafresh.com/3d-weaver/




(a) During Mission (b) Request Help (c) Return to Base Request (d) Mission Finished

Fig. 4: **Smart Watch Application:** shows the main Watch UI screens. During the mission the worker can view different quantities such as humidity, temperature inside the suit, the heart rate and the elapsed time. The participant can ask for help using the help button. Request help shows the screen when the worker requested help. Return to Base shows the UI as soon as the Commander requested a return. As soon as the worker returned to the secure zone the mission finished screen is shown. All buttons are highlighted with an orange border.

in most cases not sufficient to complete a task. In order to extend that working period the deployment of smart textiles in combination with mobile monitoring and cooling possibilities offers a solution (compare Fig. 3).
Currently, monitoring systems are hard-wired systems using sensors connected to a base unit in a shoulder bag. They can measure body temperature, the humidity in the suit and the heart-rate of the worker. An expert, the medical advisor, attaches the sensors to the worker. Each sensor is currently taped to the appropriate area on the body of the worker. During the mission a real-time monitoring is not possible as data-loggers are used. The doctor totally relies on his experience estimating the condition of the worker. The commander is the decision maker, deciding when a worker has to end a task. Consulting the doctor is inevitable for him. To the best of our knowledge, a comprehensive system that combines real-time monitoring, feedback channels between all stakeholders as well as context/condition dependent cooling is not available for heavy duty operations. The described system is our first vertical prototype showing the feasibility of the system design focusing on the software architecture. We conducted first experiments using 3D printing techniques in combination with smart textiles. We transferred knowledge from the domain of health monitoring to realize CHEST for heavy duty, harsh environment applications.

## 4. Visionary Scenario

CHEST focuses on professional, harsh environments where workers are forced to use CBRN-Suits. Fig. 2 illustrates the visionary scenario during a contamination mission. Within the scenario, 3 actor roles are defined. The **Worker**, the **Medical Advisor (MA)** and the **Commander**. The worker receives instructions by the mission commander to fulfill the mission tasks. The MA is remotely monitoring the worker. All three actors are able to communicate with each other using an applicable communication system.
The environment where the scenario takes place is divided into two zones: the *secure zone* and the *danger zone*. The *secure zone* is the home base of the mission and is defined as safe environment. The base station is located within this zone. It consists of a container equipped with a wireless identification technology, that forms a geofence or halo around this area. Within the container all the infrastructure and equipment that is needed to monitor, control and communicate is located. Within the *danger zone* the worker has to operate. It is a harsh, hazardous environment, where special clothing such as CBRN-Suits are necessary.
Before the mission starts, the **Workers** are heading to the **Commander**. The **Commander** selects the **Workers** needed for the task, coordinates and plans the mission. As soon as the **Commander** has selected all the **Workers** needed for the mission, each **Worker** has to pick a CBRN-Suit from the stock. Each **Worker** is equipped with a personalized mobile device that is able to scan an identifier on each CBRN-Suit. After the identifier has been scanned the suit is mapped to the **Worker** who scanned it. As soon as the **Worker** is ready the mobile device is used to send the status to the **Commander**. The **Commander** receives a message in a browser application, which **MA** is ready to perform the task and arrange the group accordingly. An avatar represents each **Worker**. Additionally, a color-coded border indicating the technical integrity of the equipment surrounds each avatar. The **Commander** starts the mission and selects a MA, that is responsible for the health condition of each worker.
The **MA** in a separate part of the container can instantly see an anonymized view of all selected workers within the mission. The **MA** has no information about the CBRN-suit-worker mapping. Due to legal constraints the **MA** must not know about the identity of the **Workers** and must not receive any information that allows a person-data mapping.

The **Workers** are now ready to start the mission. They are heading to the *danger zone*, operating and working on their task. The **MA** is able to monitor each participant and the system aggregates an overall status (stresslevel). In case a situation appears where a **MA** is entering a critical health situation, the **MA** can inform the **Commander**. The **Commander** receives a notification within the browser application. The **Commander** now decides if the **Worker** has to leave the *danger zone*. In case a **Worker** needs to leave the corresponding **Worker** should receive a notification on his wearable wrist device. Such a notification can be a vibration of the wearable device or an acoustic signal. As soon as the **Worker** returns to the *secure zone*, the CBRN-Suit can detect the halo of the *secure zone* and the **Commander** can see in his browser app that the corresponding **Worker** has arrived in the *secure zone*. In the following paragraphs we describe concrete visionary scenarios.

**Decontamination of Vehicles - Monitor and Control:** We assume that a worker needs to decontaminate several vehicles. Approximately, the task will take up to 1.5 hours. The MA that monitors the medical status of each worker needs to guarantee that the worker is still capable of working without risking health and therefore the safety of the worker. The MA needs to have a live monitoring system that is able to monitor how the workers' physical condition is. Therefore, the MA can monitor all data and even control the CBRN-suit remotely, namely airflow, heartrate and temperature. These are values that need to be controlled in quasi real-time if a dangerous situation occurs.

**Group Safety:** A group of workers are operating in a harsh environment within the danger zone. It is of major importance that each member during a mission knows about a potential risky situation of other group members. Therefore, each CBRN-Suit transmits critical conditions to the group members and to a monitoring system. In case of a potential collapse of a group member, everybody needs to know about this potential safety risk. They might help or support the other team member.

**Web Based Consulting of Medical Advisors:** In case of a potentially dangerous or critical situation the medical doctor might want to consult other medical advisors. The system should be able to create a shareable view of the current participant that other doctors can estimate if they come to the same conclusion.

**Documentation:** After a mission has finished all commands, actions and all data are stored for documentation reasons. As soon as they are stored, stakeholders from the administrative layer can be informed.

## 5. Subsystem Decomposition

The communication between all subsystems is realized using a publish/subscribe mechanism in combination with RESTful services. The architecture includes a MQTT Broker architecture that CHEST is relying on. CHEST is decomposed into five subsystems, the Health Monitoring, the Smartphone, Smartwatch, the Monitoring Server and the Browser application.

The Health Monitoring System can be further subdivided into the *Smart Textile* and the *Suit Sensors*. The *Smart Textile* is capable to monitor the heart rate and the Heart Rate Variability (HRV) as accepted indicator to monitor health condition [13]. The *Smart Textile* integrates the ECG sensors and a techunit that is collecting the vital data. It provides the data via Bluetooth low energy (BLE). In this prototype we used the ambiotex[9] shirt and cooperated with the manufacturer. The *Suit Sensors* are able to monitor the interior physical quantities such as temperature and humidity. We used a BeeWi[10] Weather Station as a first start. It also provides all measured data via BLE.

The *Smartphone* application is collecting the data from the *Health Monitoring* subsystem via BLE. It is capable of registering the worker and serves as center peace of data aggregation and communication. As communication channel we use WiFi at 2,4 GHz. Moreover, the subsystem can sense nearby BLE beacons that are used to detect the secure zone. It provides information for the *Smartwatch*.

The *Smartwatch* serves as information display. It is able to provide vital data and instructions triggered by the MA or the Commander. It provides acoustic and vibration feedback to the worker. In Figure 4d the screens for the vital information, the help request, the command to return to the base and the mission finished screen is shown.

The *Monitoring Server* subscribes to special topics at the MQTT Broker. The topic structure is based on the roles described in Section 4. To estimate technical integrity the *Smartphone* provides a heartbeat mechanism to indicate the technical integrity of the system. The *Monitoring Server* stores all data during a mission. It provides a RESTful API to access historical data and specific time interval based data. Following the publish/subscribe mechanism a remote quasi real-time monitoring is possible.

The Browser provides a User Interface for the MA and the Commander. It provides all relevant data in a dashboard kind of UI Design. Figure 5 shows the MA Dashboard. In the highlighted section (1) anonymized information about the participant can be seen. At (2) color coded indicators of the technical parameters can be seen. The MQTT Connection, the CBRN-Suit connection as well as the sensor connection can be checked. At (3) All measured values such as the heartrate, the temperature, the humidity as well as the heartrate variance can be monitored. The stresslevel can serve as an overall indicator about the current situation of the worker. In (4) historical data and sharing possibility controls are provided. Also Feedback possibilities are provided as

[9] Ambiotex `www.ambiotex.com`
[10] BeeWi Weather Station

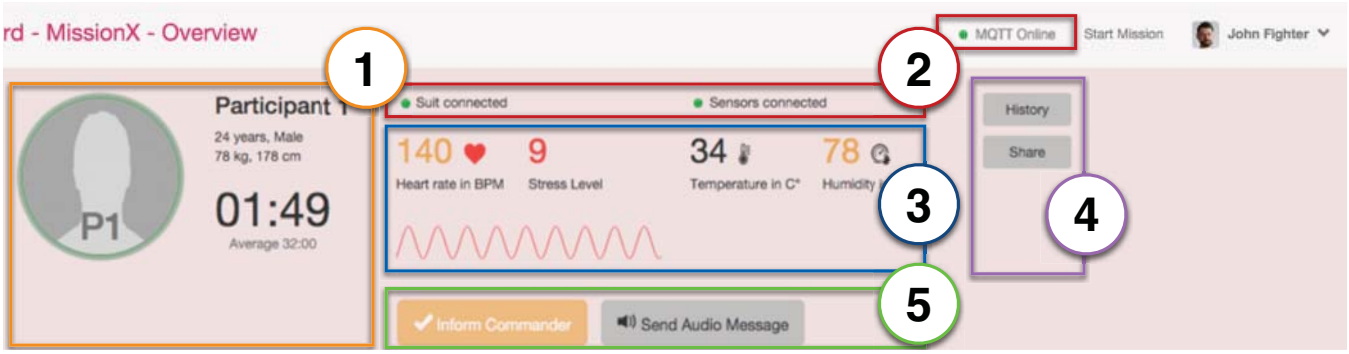



Fig. 5: **Medical Advisor Dashboard:** The screenshot shows the 5 different parts of the MA Dashboard: (1) the anonymized worker information, (2) technical status parameters, (3) all gathered values, (4) historical data and sharing possibility controls and (5) feedback channel controls.

shown in (5).

## 6. Limitations and Future Work

CHEST faces several limitations we are currently trying to address. The cooling unit needs to be integrated into the suit using the already developed 3D printed cooling structures. The sensor quality and medical effectiveness needs to be evaluated. The data transmission and the security of the communication channel need to be addressed.

## 7. Conclusion

In this paper we describe CHEST, a Human-Centric Cyber-Physical System. It is a vertical prototype that is designed to monitor and reduce physical stress of workers in harsh, hazardous environments. We could show the technical feasibility and major benefits compared to data logger based systems. We brought together researchers from different domains all contributing with their domain specific knowledge. CHEST supports the process of setting up a mission-team using mobile devices. CHEST allows remote health monitoring, in combination with wearables and smart textiles. Moreover, it includes a feedback channel between all included stakeholders, namely, the worker, the Medical Advisor and the Mission Commander. It includes a context aware secure zone detection so that the Mission Commander knows, which worker returned to the secure zone. We are currently extending the prototype to offer smart cooling capabilities, directly integrated into the system.

## Acknowledgments

Thanks to Aly Saleh, Dimitar Magurev, João Trindade, Josef Seidl, Konstantin Kromer, Richard Otto, Simon Zimmermann and Valeriia Chernenko for their great effort during the project. Special thanks to Florian Dennerlein for the constructive cooperation and support.

## References

[1] J. Reason. *Human Error*. Cambridge University Press, 1990.
[2] Jean-Claude Laprie. Dependable computing and fault-tolerance. *Digest of Papers FTCS-15*, pages 2–11, 1985.
[3] M. Sanders and E. McCormick. *Human Factors In Engineering and Design*. McGraw-Hill international editions: Psychology series. McGraw-Hill Education, 1993.
[4] J. Von Knop. *A Process for Developing a Common Vocabulary in the Information Security Area*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2007.
[5] O. Goloubeva, M. Rebaudengo, M.S. Reorda, and M. Violante. *Software-Implemented Hardware Fault Tolerance*. Springer US, 2006.
[6] Raija MT Laukkanen and Paula K Virtanen. Heart rate monitors: state of the art. *Journal of sports sciences*, 16:3–7, 1998.
[7] Sukun Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium*, pages 254–263, April 2007.
[8] A. Pantelopoulos and N.G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(1):1–12, Jan 2010.
[9] Yang Hao and Robert Foster. Wireless body sensor networks for health-monitoring applications. *Physiological measurement*, 29(11):R27–R56, 2008.
[10] P.J. Soh, G.A.E. Vandenbosch, M. Mercuri, and D.M.M.-P. Schreurs. Wearable wireless health monitoring: Current developments, challenges, and future trends. *Microwave Magazine, IEEE*, 16(4):55–70, May 2015.
[11] C.W. Mundt, K.N. Montgomery, U.E. Udoh, V.N. Barker, G.C. Thonier, A.M. Tellier, R.D. Ricks, B.B. Darling, Y.D. Cagle, N.A. Cabrol, S.J. Ruoss, J.L. Swain, J.W. Hines, and G.T.A. Kovacs. A multiparameter wearable physiologic monitoring system for space and terrestrial applications. *Information Technology in Biomedicine, IEEE Transactions*, 9(3):382–391, Sept 2005.
[12] K.J. Glitz, U. Seibel, U. Rohde, W. Gorges, A. Witzki, C. Piekarski, and D. Leyk. Reducing heat stress under thermal insulation in protective clothing: microclimate cooling by a 'physiological' method. *Ergonomics*, 58(8):1461–1469, 2015. PMID: 25679096.
[13] Juul Achten and Asker E Jeukendrup. Heart rate monitoring. *Sports medicine*, 33(7):517–538, 2003.

# Thermal-Aware Task Allocation and Scheduling for Heterogeneous Multi-core Cyber-Physical Systems

**Shikang Xu, Israel Koren and C. M. Krishna**
Department of Electrical and Computer Engineering University of Massachusetts Amherst, Amherst, MA, 01003

**Abstract**— *Over the past few years, the impact of heating on device failure rates has become increasingly important in cyber-physical systems. Such systems are often used in harsh environments and have weight, volume and power constraints which make it hard to dissipate heat economically and effectively. Thermal-aware task scheduling techniques are therefore called for to reduce heat stress on the computational platform.*

*In this paper, we outline a thermal-aware task allocation and scheduling heuristic for use in computational platforms that feature heterogeneous computational cores. Such systems, consisting typically of some powerful, out-of-order, cores, together with simple, in-order, cores, allow the user a wider range of power-performance tradeoffs than traditional homogeneous multicore systems. We show how using fairly simple thermal-aware task allocation and scheduling principles results in a substantial enhancement in the expected lifetime of the system.*

***Index Terms*— *DVFS, Real-time systems, Cyber-physical systems, Lifetime extension, Heterogeneous multi-core***

## 1 Introduction

Cyber-physical systems (CPSs) are proliferating in life-critical and cost-sensitive contexts today. The economical provision of high reliability in such systems is very important.

Thermal stress is an important source of accelerated device aging, resulting in premature device failure. Indeed, failure rates are often modeled as exponentially increasing with absolute temperature. Furthermore, much of the thermally-induced damage is cumulative, not transient. As technology has advanced and feature sizes have shrunk, heat density has increased, making high temperature an important factor in device degradation and failure.

We focus in this paper on how to allocate and schedule CPS tasks on heterogeneous multi-core platforms, in a thermal-aware fashion. Such platforms, such as big.LITTLE from ARM [1], typically consist of high-performance cores with advanced pipelining features for enhanced throughput, coupled with very small cores only capable of simple, in-order, instruction processing. All cores, simple and complex, on the platform share the same instruction set so that task migration across core types is possible. Such platforms are attractive for CPS since they offer an increased flexibility in scheduling dynamically varying real-time workloads [2]–[4].

In this paper, we introduce a simple task allocation and scheduling algorithm for real-time systems using a heterogeneous multi-core platform. The aim is to meet all deadlines while reducing thermal stress. Our simulations show that this algorithm significantly enhances the core Mean Time To Failure (MTTF) over a baseline algorithm.

The rest of the paper is organized as follows. Section 2 reviews related prior work and points out the thermal impact on lifetime and reliability. Section 3.2 contains our algorithm. Simulation results illustrating the performance of this algorithm are presented in Section 4. Section 5 concludes this paper.

This work was partially supported by the National Science Foundation under grant CNS 1329831.

## 2 Background

### 2.1 Related Works

Over the past decade, much work has been conducted on general-purpose, single-ISA, heterogeneous multi-core systems task scheduling [5]–[10]. In [9], Craeynest *et al.* developed a workload-to-core mapping algorithm for single-ISA heterogeneous multi-core processors to improve performance by dynamically monitoring the behavior (through measures like Cycles per Instruction and Instruction-Level Parallelism) of tasks. In [10], Zhang *et al.* proposed an algorithm based on dynamic execution behavior of workloads, to map tasks to cores to achieve higher energy-efficiency while maintaining comparable performance as in [9].

For real-time systems, recent research focuses on algorithms to guarantee an increased amount of workload to be finished before its deadline. Baruah introduced a polynomial-time feasibility analysis method that can determine if a set of real-time tasks can be scheduled on an multi-core system and tasks can migrate between cores [2]. Kim *et al.* proposed a scheduling algorithm for aperiodic hard real-time tasks executing on heterogeneous platforms that restrict tasks to be executed on certain cores according to their expected completion time [11].

Raravi *et al.* showed that, if the tasks can only migrate between the cores of the same type, an optimal algorithm needs cores that are $1 + \frac{\beta}{2}$ times faster ($0 < \beta < 1$) than in a system where tasks can migrate among multiple core types [3].

Recently, Chwa *et al.* proposed a task scheduling algorithm for real-time systems with two core types [4]. In their algorithm, the cores are assumed to share the same instruction set architecture (ISA) so that tasks can migrate between core types. A two-phase approach is followed: first, assignment of tasks to the appropriate core type, and second, a modified DP(deadline partitioning)-fair [12] method is used to schedule the workload on each type.

### 2.2 Thermal Related Reliability and Lifetime

The reliability of VLSI circuits is affected by multiple mechanisms. Modeling these has been an active research topic for decades. Electromigration (EM) and Oxide Breakdown are reported to be dominant permanent failure mechanisms of VLSI circuits as CMOS technology scales [13] and consequently, these are the modes that we focus on.

According to [14], the Mean-Time-To-Failure (MTTF) due to the oxide breakdown process is given by:

$$MTTF_{bd} = A_{bd} * V^{-(a-bT)} * e^{\frac{X+(Y/T)+ZT}{kT}} \quad (1)$$

where $V$ is the voltage applied to the gate oxide, $T$ is the absolute temperature in Kelvin, $k$ is Boltzmann's constant and $A_{bd}$ is a scale factor. The values of the other parameters are [14]: $a = 78$, $b = -0.0081$, $X = 0.759eV$, $Y = -66.8eV * K$ and $Z = -8.37e4eV/K$. Note that these parameters, or even the MTTF model, can be different as the CMOS manufacture technology developing.

The mechanism behind EM has been studied extensively. One of the early results that is still widely used for estimating the MTTF due to EM, was proposed by Black [15]:

$$MTTF_{em} = A_{em} * J^{-n} e^{\frac{E_a}{kT}} \quad (2)$$

where $A_{em}$ is a scale factor, $J$ is the current density, $E_a$ is activation energy and $n$ is a material based constant. For copper, these values are $J = 1e^6 A/cm$ [16], $E_a = 0.9eV$ and $n = 1.1$ [17].

The failure of a system is a random process and the reliability of a system at time $t$ is the probability that the system is functional throughout the time interval $[0, t]$. The probability of a device failure occurrence if often modeled by by the Weibull distribution [18]:

$$F(t) = 1 - R(t) = 1 - e^{-(t/\alpha)^{\beta}} \quad (3)$$

where $F(t)$ is the failure occurrence probability, $R(t)$ is the reliability function, $\beta$ is the Weibull slope parameter ($\beta$=2, [19]), and $\alpha$ is a scale parameter satisfying $\alpha = MTTF/\Gamma(1+1/\beta)$.

The reliability expressions shown above were obtained from a combination of experimental results and proposed physical models. In this paper, the approach of [18] is adopted to calculate the reliability in a dynamic thermal environment (i.e., under varying temperatures). Time is divided into $k$ time frames, [0,$\Delta$t],[$\Delta$t, 2$\Delta$t], ..., [($k$-1)$\Delta$t, $k\Delta$t]. Each time frame is short enough so that the temperature and voltage are roughly constant within it. The resulting reliability of a functional block in an IC (e.g., an integer execution unit), denoted by $R_{blk}(t)$, is:

$$R_{blk}(t) = R(k\Delta t) = \prod_{i=1}^{i=k} [1 - (R_i((i-1)\Delta t) - R_i(i\Delta t))] \quad (4)$$

where $R_i(i\Delta t)) = R_{i_{em}}(i\Delta t)) * R_{i_{bd}}(i\Delta t))$; $R_{i_{em}}(i\Delta t))$ and $R_{i_{bd}}(i\Delta t))$ are the reliabilities due to electromigration and oxide breakdown, respectively, and are calculated by Equation 3 using the MTTFs derived from the reliability model using the temperature of the $i$th time interval.

The reliability of a core at time $t$ is the product of the reliability of all the functional blocks of the core at time $t$.

## 3 Thermal Aware Heterogeneous Multi-core Scheduling Algorithm

### 3.1 System Models

The heterogeneous multi-core system we consider consists of two core types: out-of-order ("big") and in-order ("small") cores. The former are much more complicated, with sophisticated pipelining techniques to improve performance (at the cost of power consumption); the latter are simple and slow. All cores use the same instruction set architecture, meaning that tasks can, deadlines permitting, execute on any core. All cores of the same type are identical to one another. As already mentioned, this structure is commercially realized these days, one example being the ARM big.LITTLE architecture.

The task set is periodic, with deadlines equal to the period. This is a common model for real-time systems in practice [20]; we will later extend it to the case where deadlines are constrained to be less than or equal to their period. Each task $\tau_i$ is characterized by the following three parameters: period ($p_i$), Worst-Case Execution Time (WCET) on a big core ($e_i^b$) and WCET on a small core ($e_i^s$). In this paper, it is assumed that for all $\tau_i$, $e_i^b \leq e_i^s$ and the utilization on the big core satisfies $u_i^b = \frac{e_i^b}{p_i} \leq 1$. If the utilization on small core, $u_i^s = \frac{e_i^s}{p_i}$, is larger than 1, only using small cores is insufficient to meet the worst-case computational demands of the workload. Tasks can be migrated from one core to another during execution.

We also assume that the cores share the main memory and last level cache (LLC). With this assumption, the task migration overhead is negligible [9]. For systems with many-cores, the assumption may not be true. However, cores are usually clustered into groups that share LLC and memory. The proposed algorithm can be applied to the cores that are lie in the same group.

The reliability figure-of-merit we use is the product of the individual reliabilities of all the cores. The expected lifetime of the system is defined as the point at which this figure of merit declines to a predetermined level.

### 3.2 Task Assignment and Scheduling

#### 3.2.1 Baseline Algorithm

To provide appropriate context for our work, we start with a brief description of a baseline multicore scheduling algorithm. As already mentioned, Chwa, *et al.*, propose an optimal task scheduling algorithm for heterogeneous multi-core real-time systems. Two core types are assumed [4]. Optimality is demonstrated by proving that as long as a task set can be feasibly scheduled on a heterogeneous multi-core system, it can be scheduled by their algorithm. The algorithm proposed in [4] includes two phases: workload assignment (Hetero-Split) and schedule generation (Hetero-Wrap). A high-level description is shown in Figure 1. In [4], it is assumed that some tasks will execute faster on type-I cores while other tasks will execute faster on type-II cores. If the utilization of a task $\tau_i$ on the cores that executes it slower can be greater than 1, there is a minimum fraction of such a task that must be executed on

**Notation:**
$e_i^1/e_i^2$: WCET of $\tau_i$ on type-I/II core
$\Gamma^1/\Gamma^2$: list of tasks on type-I/II core
$x_i^1/x_i^2$: portion of $\tau_i$ on type-I/II core
$M_1$: list of tasks $u_i^1 * x_i^1 + u_i^2 * x_i^2 = 1$ and $x_i^1 > 0, x_i^2 > 0$
$M_2$: list of tasks $u_i^1 * x_i^1 + u_i^2 * x_i^2 < 1$ and $x_i^1 > 0, x_i^2 > 0$
$P_1/P_2$: list of tasks only assigned to type-I/II cores
**Hetero-Split:**
calculate minimum fraction of each task $i$ on the faster core using equation (5) and (6)
$\Gamma^1 \leftarrow \{\tau_i | e_i^1 < e_i^2\}$, $\Gamma^2 \leftarrow \{\tau_i | e_i^2 < e_i^1\}$
IF workload on type-I (type-II) cores is larger than its capacity
**repeat**
find the $\tau_k$ with the smallest (largest) $\frac{e_i^1}{e_i^2}$
move $\tau_k$ to type-II (type-I) cores ($lo_k^1$ ($lo_k^2$) cannot be moved)
**Until** workload on type-I (type-II) cores is no larger than its capacity

**Hetero-Wrap**:
For tasks in $M_1$, $M_2$ and $P_1$
fill the type-I cores' slices from the beginning of slice
For tasks in $M_1$, $M_2$ and $P_2$
fill the type-II cores' slices from the end of slice

Fig. 1. General flow of the algorithm in [4]

the other core type. This minimum fraction (denoted by $lo_i$) is calculated in the following way:

$$lo_i^1 = \begin{cases} \frac{u_i^2 - 1}{u_i^2 - u_i^1}, & \text{if } u_i^2 > 1 \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

$$lo_i^2 = \begin{cases} \frac{u_i^1 - 1}{u_i^1 - u_i^2}, & \text{if } u_i^1 > 1 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where $u_i^1(u_i^2)$ is the utilization on Type I (Type II) core.

In the task assignment phase, the workload is assigned to the different core types. The first step is to determine the minimum portion of each task that needs to be executed on the type of core that executes it faster and allocate that portion to such a type. If the workload assigned to any type of core is more than its ability to schedule after such an assignment, then the algorithm cannot schedule the task set. The second step will initially assign the rest of the workload (tasks with utilization smaller than 1 on both types cores and tasks with utilization larger than 1 on one type of core excluding the portion dealt with in first step) to the type of core that can execute it faster. If there is a type of core that cannot finish its assigned workload on time, the task will then be moved to the other core type. The order at which tasks are moved is determined by the ratio of execution times on the two core types for each task, e.g., when moving from type-I to type-II core, the task with the largest $\frac{e_i^1}{e_i^2}$ will be moved first. But if the moving task has a portion that must be executed on current core (from equation (5) and (6)), this portion will not be moved.

In the phase of schedule generation, the algorithm is a modification of DP-fair [12]. The execution is divided into slices and each task will be executed for an interval that equals the product of its utilization on the current core and slice length, e.g., $\tau_i$ will be executed for 5 seconds on a type-I core if the slice is 10 seconds and $\tau_i$ has a utilization of 0.5 on type-I core. The situation where a task is scheduled to be executed on two types of cores at the same time needs to be avoided. Tasks are divided into four groups: $M_1$ are the tasks where the sum of utilization on the two core types equals 1; $M_2$ are tasks where the sum of the utilization on the two core types is less than 1; $P_1$ are tasks that are only assigned to type-I cores and $P_2$ are tasks that are only assigned to type-II cores. The time-slices of type-I cores will be assigned to tasks in the order of $M_1$, $M_2$ and $P_1$ from the *beginning* of each slice. The time-slices of type-II cores will be assigned to tasks in the order of $M_1$, $M_2$ and $P_2$ from the *end* of each slice.

### 3.2.2 Thermal-Aware Task Allocation and Scheduling

The baseline algorithm described above does not take thermal considerations into account. We describe here a thermal-aware task allocation and scheduling algorithm, which we will then show has significant lifetime-enhancement benefits. The system model assumed was described in Section IIIA, namely, we assume "smaller" and "bigger" cores. Smaller cores do not have performance-enhancing (but power-hungry) features like out-of-order execution.

The general principle behind this algorithm is to preferentially assign load to the smaller cores; only that portion of a task that cannot feasibly be executed on the smaller cores is assigned to the bigger core. Second, we use offline task profiling to obtain the average temperature ($T_{avg}^i$) when running a task on a big core. When some tasks have to be assigned to big cores, instead of choosing the task with smallest $\frac{e_i^b}{e_i^s}$ as in [4], we choose the task with smallest $T_{avg}^i * \frac{e_i^b}{e_i^s}$. (Note: this modification will potentially sacrifice optimality.)

The pseudo code of the thermal-aware task assignment is shown in Figure 2. In stage 1, the minimum fraction of each task that needs to be executed on big cores is allocated to big cores. If the big cores cannot execute the allocated minimum workload, the given task sets cannot be scheduled, and the algorithm returns failure. In stage 2, all the remaining workload is assigned to small cores. In stage 3, if the small cores cannot schedule the assigned workload, some workload needs to be moved to the big cores. In the workload reassignment, the task $\tau_k$ with the smallest product of $\frac{e_i^s}{e_i^b}$ and the average execution temperature on the big core will be moved first. If the small cores still cannot schedule the workload on them after moving the whole task, the whole $\tau_k$ will be moved to big cores. Otherwise, the algorithm only moves that part of the task such that the small cores can schedule the workload on them. Once the task assignment to cores is successfully determined according to this process, the Hetero-wrap scheduling procedure from the baseline algorithm is used to generate the task schedule.

## 3.3 DVFS Algorithm

Dynamic Voltage and Frequency Scaling (DVFS) has long been used to reduced the thermal impact on processors. In

**Notation**
$u_i^b/u_i^s$: utilization of $\tau_i$ on big/small core
$x_i^b/x_i^s$: fraction of $\tau_i$ assigned to big/small core
$lo_i$: minimum fraction of $\tau_i$ assigned to big core
$y_i^b/y_i^s$: fraction of $\tau_i$ assigned to big/small core excluding $lo_i$
$\Gamma_b/\Gamma_s$: list of tasks on big/small core
$m_b/m_s$: number of big/small cores
$er_i$: $T_{avg}^i * \frac{e_i^s}{e_i^b}$ of $\tau_i$

**Task_ alloc($T$)**
Stage 1:
Allocate $lo_i$
**if** $\sum lo_i * u_i^b > m_b$
  return not feasible
Stage 2:
**for** all $\tau_i$ in $T$
  $\Gamma_s \leftarrow \Gamma_s \cup \tau_i$
  $y_i^s \leftarrow 1 - lo_i$
Stage 3:
**if** $\sum y_i^s * u_i^s \leq m_s$
  return $\{x_i^b | x_i^b = lo_i\}$,$\{x_i^s | x_i^s = y_i^s\}$
**else**
  **while** $\sum y_i^s * u_i^s > m_s$
    find $\tau_k$ with smallest $er_i$ in $\Gamma_s$
    **if** $\sum y_i^s * u_i^s - y_k^s * u_k^s > m_s$
      $y_k^b \leftarrow 1 - lo_k$
      $y_k^s \leftarrow 0$
      $\Gamma_s \leftarrow \Gamma_s - \tau_k$
    **else**
      $y_k^b \leftarrow \frac{\sum y_i^s * u_i^s - m_s}{u_k^s}$
      $y_k^s \leftarrow 1 - lo_k - y_k^b$
    **if** $\sum y_i^b * u_i^b > m_b - \sum lo_i * u_i^b$
      return not feasible
return $\{x_i^b | x_i^b = lo_i + y_k^b\}$,$\{x_i^s | x_i^s = y_i^s\}$

Fig. 2. Thermal Aware task assignment

**Algorithm:IPC_DVFS**
**Notation:**
$T_{sys}$: Tasks in system
$P$: list of partitioned tasks on core
$t_{sys}$: system time, initialized at 0
$f_{high}/f_{low}$: processor high/low frequency level
$\Delta t$: time step
$t_{slice}$: length of scheduling slice
$F$: flag array indicating if task $i$ has finished the current iteration
$IPC$: Array recording the IPC of $\tau_i$ in the previous time step
$SA$: schedule array for core in each slice after task assignment

**At every time step**
For each $\tau_i$ in system
  IF $\tau_i$ finished in past $\Delta t$
    $F[i] = 1$
  IF $t_{sim}$ mod $p_i$=0
    $F[i] = 0$
IF $t_{sim}$ mod $t_{slice}$=0
  For all $\tau_i$ in $T_{sys}$
    IF $F[i]$=0
      $T \leftarrow T \cup \tau_i$
  $Task\_Alloc(T)$
  Generate SA for each core
  $Alloc\_Slack()$
  Set count=0 on each core
On each core
  execute $\tau_{SA[count].index}$ at speed SA[count].speed
  set IPC[SA[count].index] to IPC in the $\Delta t$
  IF $t_{sys}$ mod $t_{slice}$=SA[count].end or task finishes
    count+=1

Fig. 3. IPC based DVFS

[21], a DVFS scheme based on Instructions-Per-Clock (IPC) monitoring is presented to improve the lifetime of processors by preferentially using slack to slowdown a high-IPC phase of workload. When executing tasks according to the slice schedule generated using the method described above, DVFS is applied using the concept proposed in [21].

As stated above, there are partitioned tasks and migrating tasks on cores. Applying DVFS on migrating tasks may result in executing the same task on different cores at the same time, which is not allowed. Thus, in this paper, only partitioned tasks are subject to slowdown by voltage scaling. The pseudo code of the DVFS algorithm is shown in Figures 3 and 4. In Figure 3, at the beginning of every time step, the tasks completed in the prior time step are indicated using a finishing flag array. If a new iteration of a task is released, this flag is reset. Also, at the beginning of each time slice, the schedule is rearranged based on current uncompleted tasks in the system; the available slack in the coming time slice is then calculated. The initial task assignment and scheduling is based on the Worst Case Execution Time (WCET) of each task. If a task finishes earlier than its WCET, there is no need to allocate time for this task in the coming time slices until its next iteration is released. Thus, at the beginning of every time slice, tasks that have not finished will be reassigned using the task assignment algorithm described in the previous section. Then, the slack will be used to slow down partitioned tasks on big cores using the algorithm shown in Figure 4.

On big cores, the partitioned tasks are arranged to be executed first in the coming time slices. Firstly, all the uncompleted partitioned are clustered to the front part of the slice and all the uncompleted migrating tasks are clustered to the end part of the slice. The gap between these two clusters is the available slack. As is shown in Figure 4, the IPC of each task in the prior time step is recorded. The available slack is then allocated to the task with the highest IPC. Based on how much slack is available, the task with highest IPC in a given time slice is either wholly or partially slowed down. If there is still some slack left after the above process, the task with the second highest IPC will go through the same process. This process continues until the slack is used up or all partitioned tasks are slowed down.

**Notation**
$P$: list of partitioned tasks
$f_{high}$/$f_{low}$: high/low frequency level of core
$SA[i].index$: index of task that is at position i of array SA
$SA[i].start$: start time of the task at position i
$SA[i].end$: end time of the task at position i
$SA[i].speed$: frequency level to run task position i

**Alloc_Slack()**
On each big core
$\Gamma_p$={$SA[i]|SA[i].index \in$P}
front=0, end=$t_{slice}$
FOR $i$ in (0..SA.size-1)
  IF $SA[i] \in \Gamma_p$
    $l$=SA[i].end-SA[i].start
    SA[i].start=front
    SA[i].end=front+$l$
    front=SA[i].end
FOR i in (SA.size-1..0)
  IF $SA[i] \notin \Gamma_p$
    $l$=SA[i].end-SA[i].start
    SA[i].end=end
    SA[i].end=end-$l$
    end=SA[i].start
$slack$=end-front
FOR SA[i] in $\Gamma_p$
*//access in IPC in past $\Delta t$ descending order*
  l=SA[i].end-SA[i].start
  sn=l*$\frac{f_{high}-f_{low}}{f_{low}}$
  *//slack needed to slowdown the how part of SA[i]*
  IF slack>sn
    SA[i].speed=LV
    SA[i].end+=sn
    rearrange SA[k]s following SA[i] in SA
    slack-=sn
  ELSE IF slack>0
    sp=slack*$\frac{f_{low}}{f_{high}-f_{low}}$
    *//workload can be slowed by slack*
    SA[i].speed=LV
    SA[i].end=SA[i].start+slack+sp
    insert (index=SA[i].index, speed=LV, start=SA[i].end, end =start+l-sp) to SA
    rearrange SA[k]s following SA[i] in SA
    slack=0
  ELSE break

Fig. 4. Slack allocation on big core

# 4 Experimental Results

## 4.1 Experiment Setup

A system with one out-of-order (big) core and one in-order (small) core is simulated to compare the thermal-aware algorithm introduced in the previous section to the baseline algorithm. The big core has two frequency levels: 2.0 GHz and 1.2 GHz. Applications from the Mibench benchmark suite [22] are used as a workload of independent tasks. Task sets are generated by selecting five tasks randomly from the benchmark suite. The WCET for each task running on big/small core of each task is found by using the Gem5 [23] out-of-order/in-order simulator. The actual execution time of each iteration is a random number; it is selected to be normally distributed with mean equal to half the WCET and standard deviation of 0.2 of the mean, conditioned to be no greater than the WCET value.

To obtain the power consumption of the workload we used Gem5 and MaPAT [24]. These power traces were then used to generate temperature traces for each of the cores; we used the fast thermal simulator TILTS [25] which is a modification of HotSpot [26]. When a task has an actual execution time that equals $r_i$ of its WCET, its power file will be compressed in the time domain to this fraction. This ensures simulation of all the high/low power phases in the execution of a task.

The reliability of each core (thus the system) is initially set to 1. Recall that our figure-of-merit (FOM) is the operating time before the reliability of the system (i.e., the product of the individual core reliabilities) reaches a given threshold; we illustrate here the lifetimes when the FOM thresholds are $1-10^{-6}, 1-10^{-7}$, and $1-10^{-8}$, respectively. Corresponding lifetimes are evaluated for the task schedule generated by the baseline algorithm and the improvement of our algorithm over the baseline is calculated.

A simple partitioned algorithm is also introduced to be compared with the algorithm proposed in this paper. Using the partitioned algorithm, each task is statically assigned to core and can only be executed on that core. On each core, earliest deadline first (EDF) [20] is used to schedule tasks.

## 4.2 Numerical Results

Recall the main steps we used to enhance reliability in this algorithm: (A1) Assign workload to small cores as much as possible; (A2) When some tasks need to be executed on big cores, tasks that execute faster and have a lower thermal impact on big cores are assigned to big cores first; (A3) Perform workload reassignment when a task finishes before its WCET, and (A4) Use dynamic voltage and frequency scaling so long as that can be done without causing tasks to be executed on more than one core at the same time. We carry out experimental studies to evaluate the impact of these approaches on lifetime enhancement.

In Figure 5(a)-(c), the lifetime improvement without DVFS (thus using A1, A2 and A3) at different system reliabilities is shown. As already stated, the algorithm in [4] is used as baseline. "Partitioned" stands for the simple static algorithm. "A1,2" indicates that our algorithm is using A1 and A2. "A3" indicates that the algorithm is using A3. The x-axis, utilization (Worst Case), is the utilization of the task sets on the big core when the execution times of tasks equal to WCET. As shown in Figure 5, when the utilization is low, all algorithms show gains over the baseline. This is because the baseline algorithm assigns all tasks to the big core while other algorithms will assign tasks to the small core. Using A3 gains as much as 27% improvement under low utilization; under these conditions, using task reassignment, there are more chances to reassign a task to the small core and ease the thermal impact on the big core. Thus, the lifetime of the system is improved greatly. At high utilization, using A1 and A2 performs better and gains more than 10% of improvement. This is because when the system is almost fully utilized, the order in which tasks are

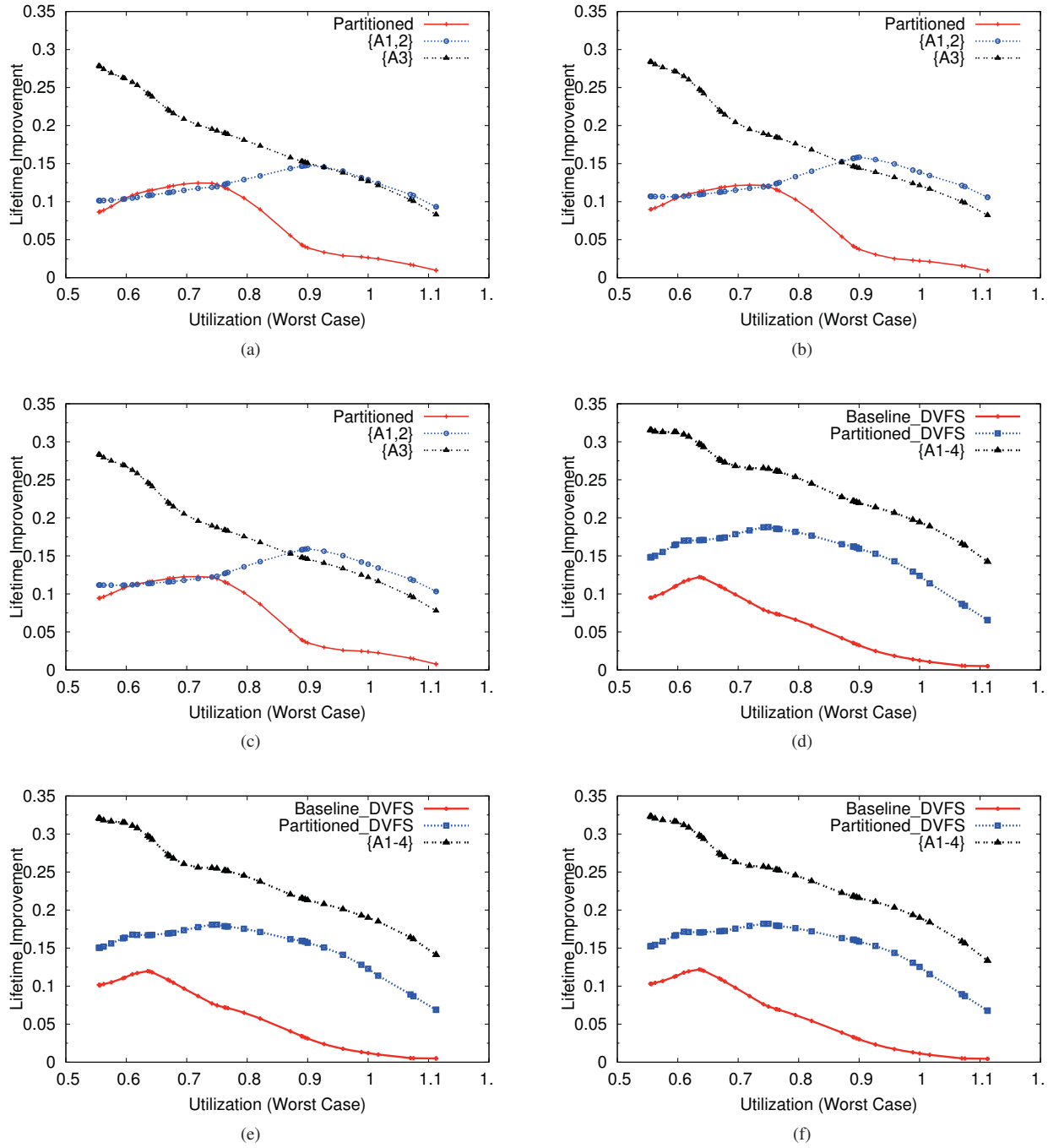



Fig. 5. Improvement for different reliability thresholds: (a) Reliability of 1-1e8 (b) Reliability of 1-1e7 (c) Reliability of 1-1e6 Improvement for different reliability thresholds with DVFS: (d)Reliabiilty of 1-1e8 (e) Reliability of 1-1e7 (f) Reliability of 1-1e6

assigned can more effectively control the thermal impact on cores; note that under these conditions, there will not be much slack for reassignment with dense workload.

Figures 5(d)-(f) show the improvement at different system reliabilities when voltage scaling is used. "Baseline_DVFS" stands for the baseline algorithm with DVFS. The baseline algorithm in [4] does not use DVFS . We add the DVFS capability by evenly assigning the slack generated by early completion to uncompleted tasks that only executed on the big core. "Partitioned_DVFS" uses the DVFS algorithm in [27]. "A1-4" uses all the 4 techniques and the algorithm we proposed in this paper. As is shown in Figures 5(d)-(f), using the techniques and algorithm proposed in this paper, the lifetime improvement is much higher than other algorithms. At high utilization, since the baseline algorithm tends to assign as much workload to the big core as possible, there is only a little slack available to apply DVFS and result in a minimal lifetime improvement. The proposed approach (using the A1-

4 techniques), on the other hand, assigns less workload to the big core, gets more slack and applies slack to tasks with higher thermal impact. These result in the the highest lifetime improvement among the three approaches. When the utilization is low, there is even more slack for DVFS on the big core using the proposed approach. Thus, the improvement over the other two is even higher.

Also note, as is shown in Figure 5, that the lifetime improvements achieved by the proposed approach at different reliabilities are almost identical.

# 5 Conclusion

Thermal issues have emerged as a key consideration in the management of cyber-physical systems. One increasingly popular architecture configuration uses high-end and low-end processing cores on a chip, sharing lower level cache and main memory. Such a mix allows the user a greater range of performance-to-power and performance-to-temperature trade-offs.

In this paper, we have introduced a thermal-aware task allocation and scheduling heuristic for use in periodic task workloads running on such heterogeneous platforms. Simulation experiments show that this heuristic provides substantial reliability benefits. In future work, we plan to extend this algorithm to cover sporadic tasks as well.

# References


[1] (ARM 2013) "big.little technology: The future of mobile. [Online]. Available: https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf

[2] S. Baruah, "Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms," in *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, Dec 2004, pp. 37–46.

[3] G. Raravi, B. Andersson, K. Bletsas, and V. Nlis, "Outstanding paper award: Task assignment algorithms for two-type heterogeneous multiprocessors," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, July 2012, pp. 34–43.

[4] H. S. Chwa, J. Seo, J. Lee, and I. Shin, "Optimal real-time scheduling on two-type heterogeneous multicore platforms," in *Real-Time Systems Symposium, 2015 IEEE*, Dec 2015, pp. 119–129.

[5] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, "Single-isa heterogeneous multi-core architectures for multithreaded workload performance," in *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, June 2004, pp. 64–75.

[6] T. Li, D. Baumberger, D. A. Koufaty, and S. Hahn, "Efficient operating system scheduling for performance-asymmetric multi-core architectures," in *Supercomputing, 2007. SC '07. Proceedings of the 2007 ACM/IEEE Conference on*, Nov 2007, pp. 1–11.

[7] J. C. Mogul, J. Mudigonda, N. Binkert, P. Ranganathan, and V. Talwar, "Using asymmetric single-isa cmps to save energy on operating systems," *IEEE Micro*, vol. 28, no. 3, pp. 26–41, May 2008.

[8] D. Shelepov, J. C. Saez Alcaide, S. Jeffery, A. Fedorova, N. Perez, Z. F. Huang, S. Blagodurov, and V. Kumar, "Hass: A scheduler for heterogeneous multicore systems," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 2, pp. 66–75, Apr. 2009.

[9] K. V. Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, June 2012, pp. 213–224.

[10] Y. Zhang, L. Duan, B. Li, L. Peng, and S. Sadagopan, "Energy efficient job scheduling in single-isa heterogeneous chip-multiprocessors," in *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, March 2014, pp. 660–666.

[11] S. I. Kim, J.-K. Kim, H. U. Ha, T. H. Kim, and K. H. Choi, "Efficient task scheduling for hard real-time tasks in asymmetric multicore processors," in *Proceedings of the 12th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part II*, ser. ICA3PP'12. Springer-Verlag, 2012, pp. 187–196.

[12] G. Levin, S. Funk, C. Sadowski, I. Pye, and S. Brandt, "Dp-fair: A simple model for understanding optimal multiprocessor scheduling," in *Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on*, July 2010, pp. 3–13.

[13] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability," in *Dependable Systems and Networks, 2004 International Conference on*, June 2004, pp. 177–186.

[14] J. Srinivasan, S. V. Adve, P. Bose, S. V. A. P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *In Proc. of the 31st International Symposium on Computer Architecture*, 2004, pp. 276–287.

[15] J. R. Black, "Mass transport of aluminum by momentum exchange with conducting electrons," in *Reliability Physics Symposium, 2005. Proceedings. 43rd Annual. 2005 IEEE International*, April 2005, pp. 1–6.

[16] Z. Lu, W. Huang, M. Stan, K. Skadron, and J. Lach, "Interconnect lifetime prediction with temporal and spatial temperature gradients for reliability-aware design and run 134 time management: Modeling and applications. very large scale integration (vlsi) systems," *IEEE Transactions on*, 2006.

[17] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "Lifetime reliability: toward an architectural solution," *Micro, IEEE*, vol. 25, no. 3, pp. 70–80, 2005.

[18] C. Zhuo, D. Sylvester, and D. Blaauw, "Process variation and temperature-aware reliability management," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 580–585.

[19] E. Wu, J. Su, W. Lai, E. Nowak, J. McKenna, A. Vayshenker, and D. Harmon, "Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides," *Solid-State Electronics*, vol. 46, no. 11, pp. 1787 – 1798, 2002.

[20] J. Liu, *Real-Time Systems*. Prentice Hall, 2000.

[21] S. Xu, I. Koren, and C. Krishna, "Improving processor lifespan and energy consumption using dvfs based on ilp monitoring," in *Green Computing Conference and Sustainable Computing Conference (IGSC), 2015 Sixth International*, Dec 2015, pp. 1–6.

[22] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, Dec 2001, pp. 3–14.

[23] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[24] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, Dec 2009, pp. 469–480.

[25] Y. Han, I. Koren, and C. M. Krishna, "Tilts: A fast architectural-level transient thermal simulation method." *J. Low Power Electronics*, vol. 3, no. 1, pp. 13–21, 2007.

[26] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotspot: a compact thermal modeling methodology for early-stage vlsi design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 5, pp. 501–513, May 2006.

[27] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 89–102, Oct. 2001.

# Application of Kalman Filter to Estimate Position of a Mobile Node in Indoor Environments

Mounika S. K. Gudipati and Shivakumar Sastry
Department of Electrical and Computer Engineering
The University of Akron, Akron OH 44325-3904, USA

***Abstract*—**Indoor location estimation is an important problem for many emerging applications that involve networked embedded systems. We present an approach to estimate the location of a mobile node using a Kalman Filter. The system model comprises four anchor nodes and one or more mobile nodes. Each mobile node uses the average value of the received signal strength to each anchor node to estimate its position using an extended Kalman Filter. These data are fused with data from a local accelerometer to improve the estimate. We present experimental results to demonstrate the precision of our location estimates. In the future, this work can be extended to reduce the error in the location estimates.

***Index Terms*—Extended Kalman Filter, Position Estimation, MultiSensor Fusion.**

## 1 Introduction

Estimating the position of mobile entities is an important problem for several emerging applications in areas such as advanced manufacturing, Internet of Things, and healthcare systems. The problem is especially challenging when it must be addressed indoors and with high precision. To address this problem, we designed and carried out experiments to understand how much precision could be achieved for position estimation in indoor environments using commercial, low-power WSN devices.

The problem of localization is very well studied in the literature [1], [2], [3]. For example, the GPS system and route navigation is widely used across the world. This problem, however, remains challenging when the resolution of the localization must be high. The localization methods in the literature are based on measuring the distance between a mobile node and a fixed set of anchor nodes at known positions. The distance between the nodes can be calculated using ToA (Time of Arrival), AoA (Angle of Arrival) or RSSI (Received Signal Strength Indicator). While the ToA and AoA can provide more accurate estimates, these approaches rely on additional hardware that are expensive. It is also not clear how much resolution these approaches can offer in laboratory and manufacturing environments that are indoor.

Several reports in the literature address this problem by using the received signal strength and triangulation based on a fixed set of anchor nodes. The option to use RSSI to estimate distance, while attractive, is complicated because of the irregularity of RF propagation in the low-power regime and because the RSSI measurements are noisy.

To address the issue of noise, we utilized an Extended Kalman Filter. A Kalman Filter is a widely used recursive prediction-update based state estimator algorithm that can minimize error variance [4]. Thus, by viewing the position of the mobile node as its state, our aim was to minimize the error in the position estimates. The prediction and update steps are combined via the Kalman gain which is calculated in each step to minimize the mean-square error. Such approaches are widely-reported in the literature in several areas including robot localization,guidance and navigation and tracking[5], [6], [7], [8].

The Kalman Filter is, however, known to provide an optimal estimate of the unknown state for a linear dynamic system with Gaussian distribution. The measurements required to assess the state of the mobile node were non-linear. To cope with this challenge, we used an Extended Kalman Filter. This framework allowed us to improve the accuracy of the position estimates, further, by integrating data from accelerometers mounted on the mobile node [9]. Kalman Filters applications for fusing data have also been have also been used widely in the literature to fuse data from multiple sensors [10], [11]. Following these methods, we designed an approach to fuse the data from an accelerometer on the mobile node with the received signal strength data.

While there are several reports in the literature that are based on simulation results, we found the following reports that were based on experiments [12], [13], [14]. Thus, the main focus of this work was to design and carryout experiments to understand how much precision could be achieved for indoor position estimation when using commercial WSN nodes that provide RSSI estimates, with no additional hardware, and when the data from a local accelerometer were fused with the data from the RSSI measurements.

The remainder of this paper is organized as follows. In Section 2 we describe the problem precisely and present the design used. After describing the localization approach in Section 3, we present results from our experiments in Section 4. Finally, we present our conclusions and next steps in Section 5.

## 2 Problem Statement and Design

In this section we describe the system model, the development platform, the measurement inputs, their calibration, and how Kalman Gain was computed.

### 2.1 System Description

The system comprises a single mobile mote and four anchor motes. Each mote was equipped with wireless transceiver that

can periodically transmit data. The mobile mote transmitted data with a known transmission power to the anchor motes. The motes, located within the transmission range of the mobile mote calculate the received RSSI values. The mobile mote was also equipped with accelerometer which was used to record the mote's inertial data.

*2.1.1* ***Development Platform****:* The IRIS mote was used as the platform to carry out the experiments. This mote has an 8-bit ATMega1281 microcontroller that is optimized for low power operations and can be easily programmed using the C language. The wireless communication between nodes is based on an AT86RF230, IEEE 802.15.4 compliant, transceiver operating in 2.5GHz ISM band. The microcontroller and the transceiver exchange information via the SPI protocol. This mote was used both as the anchor nodes and as the mobile node.

*2.1.2* ***System Model****:* The state of the mobile node was modeled in terms of its position, velocity and acceleration in a two-dimensional space $\mathbf{X_k} = [\mathbf{x}, \mathbf{y}, \mathbf{v_x}, \mathbf{v_y}, \mathbf{a_x}, \mathbf{a_y}].^{\mathbf{T}}$

Due to the non-linearities in the measurements, both for RSSI and for acceleration, Extended Kalman Filter was used to minimize errors. A discrete time model for the system was formulated as

$$\mathbf{X}_k = f(\mathbf{X}_{k-1}) + \mathbf{w_k} \tag{1}$$

where $\mathbf{X}_k$ is the state vector at the time k; the state transition function, $f(.)$, was used to estimate the future state of the mobile node and $\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$ is a random variable that represented the noise with zero mean and covariance matrix $\mathbf{Q}_k$. Here,

$$f(\mathbf{X}_{k-1}) = \mathbf{A}_k \mathbf{X}_{k-1}$$

where

$$\mathbf{A}_k = \begin{bmatrix} 1 & 0 & dt & 0 & dt^2/2 & 0 \\ 0 & 1 & 0 & dt & 0 & dt^2/2 \\ 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

where $dt$ represents the time interval for each step, $k$. The covariance $\mathbf{Q}_k$ of zero-mean Gaussian noise $\mathbf{w}_k$ is given by

$$\mathbf{Q}_k = q \begin{bmatrix} dt^4/4 & 0 & dt^3/2 & 0 & dt^2/2 & 0 \\ 0 & dt^4/4 & 0 & dt^3/2 & 0 & dt^2/2 \\ dt^3/2 & 0 & dt^2 & 0 & dt & 0 \\ 0 & dt^3/2 & 0 & dt^2 & 0 & dt \\ dt^2/2 & 0 & dt & 0 & 1 & 0 \\ 0 & dt^2/2 & 0 & dt & 0 & 1 \end{bmatrix}, \tag{3}$$

where $q = \sigma_a^2$. This covariance matrix captures the effect of different factors that could cause changes in mobile node directions. The measurement $\mathbf{Z}_k$ relies on current system state

$$\mathbf{Z}_k = h(\mathbf{X}_k) + \mathbf{v}_k \tag{4}$$

where $\mathbf{v}_k \sim N(0, \mathbf{R}_k)$.

### *2.2* ***Measurement Inputs***

As already noted, RSSI and acceleration measurements were used to estimate the position of the mobile node. Both these measurements had to be calibrated before being used.

*2.2.1* ***Calibration****:* The purpose of calibration is to choose suitable parameters that correspond to the actual devices being used in the experiment.

The acceleration data were gathered using a tri-axial accelerometer, LIS344AL [15]. The signal from this device was assumed to account for acceleration, $\mathbf{a}_t$, gravity, $\mathbf{g}_t$, small sensor bias, $\mathbf{b}_{a,t}$, and measurement noise

$$\mathbf{acc}_t = \mathbf{a}_t - \mathbf{g}_t + \mathbf{b}_{a,t} + \mathbf{v}_{a,t}, \tag{5}$$

where $\mathbf{v}_{a,t} \sim N(0, \sigma^2)$. Samples were collected from the accelerometer when the mobile node was stationary. These samples were used to account for the effect of gravity and measurement errors that arise because of temperature sensitivity, improper mounting of the sensor, or other unknown issues.

The strength of the received signal is peculiar to the specific environment in which it is received. This is usually measured in units of decibels per unit distance (dBm). The Friis equation relates the power of the received signal, $P_R$, (in Watts), the power of the transmitted signal, $P_T$, (in Watts), the receiver's antenna gain $G_R$, the transmitter's antenna gain $G_T$, the signal wavelength, $\lambda$, the distance $d$ in meters and the signal propagation constant $n$ for the environment as

$$P_R = P_T \frac{G_T G_R \lambda^2}{(4\pi)^2 d^n}. \tag{6}$$

From this relationship, using a standard reference distance of 1 m, and standard conversions from Watts to dBm, we obtain

$$RSSI = -(10n \log_{10} d - A). \tag{7}$$

To calibrate RSSI measurements, we need to calculate $A$ and $n$ for the environment. We collected RSSI values by moving the mobile node over a 10 m distance in steps of 0.5m. We obtained the average RSSI value at the mobile node, for each step, using 100 samples, from each anchor node.

The average power of the RF signal in dBm was computed as $P_R = RSSI - 45$ the value of $A$ was obtained by measuring the average RSSI value with distance 1m between the anchor nodes and the mobile node. The calibrated propagation constant $n$ was computed as

$$n = -\frac{RSSI_i - A}{10 \log_{10} d_i} \tag{8}$$

where $d_i$ is the euclidean distance between anchor node $i$ and mobile node. The final value of $n$ is obtained by averaging the values obtained from above for each anchor node.

*2.2.2* ***Measurement Model****:* Measurements, $\mathbf{Z}_k$, were incorporated into the system as indicated in Equation 4; here, $k$ is the measurement step and $h(.)$ is the observation function that provides estimates for the expected measurements when the system is in state $\mathbf{X}_k$. The measurement noise vector, $\mathbf{v}_k$, was modeled as a normally distributed random variable with zero mean and covariance matrix $\mathbf{R}_k$. We set $\mathbf{R}_k$ to a diagonal matrix since we assumed that the measurement errors were independent.

$$\mathbf{R}_k = \begin{bmatrix} diag(\sigma_{a_x}^2 & \sigma_{a_y}^2 & \sigma_{dBm_{1,k}}^2 & \sigma_{dBm_{2,k}}^2 & ...\sigma_{dBm_{4,k}}^2) \end{bmatrix} \tag{9}$$

Matrix $\mathbf{R}_k$ characterizes the errors between measured and propagation based models, and inaccuracies relating to measured acceleration. Our system used RSSI measurements $\mathbf{P_i}$ obtained as described earlier. The measurements $\mathbf{a}_x$ and $\mathbf{a}_y$ contained provided additional information regarding the node's state that was fused to improve the accuracy of the position estimation.

The measurement vector for the RSSI data, $\mathbf{Z}_k^1$, and for accelerometer $\mathbf{Z}_k^2$ is given by

$$\mathbf{Z}_k^1 = \begin{bmatrix}\mathbf{P}_1 & ...\mathbf{P}_4\end{bmatrix}^T$$

$$\mathbf{Z}_k^2 = \begin{bmatrix}0 & 0 & 0 & 0 & a_x & a_y\end{bmatrix}^T$$

The observation function was derived assuming a log-normal propagation model applied to each receiver.However since the measurement is non-linear, the measurement matrix $\mathbf{H}_k$ in the correction step must be replaced by Jacobian. The observation function $\mathbf{h}(\mathbf{X_k})$ and corresponding Jacobian $\mathbf{H_k}$ are given by

$$h(\mathbf{X}_k) = -\begin{bmatrix} A - 10\eta\log_{10}(\frac{d_1}{d_0}) \\ A - 10\eta\log_{10}(\frac{d_2}{d_0}) \\ A - 10\eta\log_{10}(\frac{d_3}{d_0}) \\ A - 10\eta\log_{10}(\frac{d_4}{d_0}) \end{bmatrix}. \tag{10}$$

$$\mathbf{H}_k = \frac{\partial h^{(i)}}{\partial x_k} \tag{11}$$

### 2.3 *Kalman Gain Computation*

To estimate the state of the system $\mathbf{X}_k$ at each time step, the Kalman Filter considers a Gaussian probability density to represent the estimated state. At each time step, predicted system state $\mathbf{X}_k^{(-)}$ is calculated from the previous estimated system state $\mathbf{X}_{k-1}^{(+)}$.

If the measurement $\mathbf{Z}_k$ is available, its effect on new state $\mathbf{X}_k^{(+)}$ has to be determined. Following the approach in [**?**], we obtained

$$\mathbf{X}_k^{(+)} = \mathbf{X}_k^{(-)} + \mathbf{K}_k[\mathbf{Z}_k - \mathbf{h}(\mathbf{X_k}^{(-)})], \tag{12}$$

$$\mathbf{P}_k^{(+)} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_k^{(-)}. \tag{13}$$

$$\mathbf{K}_k = \frac{\mathbf{P}_k^{(-)}\mathbf{H}_k^T}{\mathbf{H}_k\mathbf{P}_k^{(-)}\mathbf{H}_k^T + \mathbf{V}_k\mathbf{R}_k{\mathbf{V}_k}^T} \tag{14}$$

The matrix $\mathbf{K}_k$ denotes the Kalman Gain in step $k$.

## 3 Localization Approach

In this section, we describe the localization approach; we first describe how the position was estimated using the inertial sensors and RSSI. Finally, we describe how we fuse the two estimates for position into a single estimate.

### 3.1 *Position Estimation using inertial sensors*

The tri-axial accelerometer provided linear acceleration of the mobile node along three orthogonal axis w.r.t. the frame of the mobile node. Since we measured the acceleration, we computed the position by integrating the acceleration twice over the duration when the mobile node is moving. However, the errors in the accelerometer measurements and those introduced by numerical integration tend to accumulate during time producing large errors in the estimate of the position. Further, as discussed in the previous section, accelerometers also measure the effect of gravity and this must be removed in order to obtain the real acceleration of the mobile node.

### 3.2 *Position Estimation using RSSI*

To estimate the unknown position of mobile node, at least three anchor nodes must be able to detect and measure mobile node's signal strength. Each anchor node stores its position coordinates and value of RSSI received from the anchor nodes. The distance between mobile node and each anchor node is calculated from the measured RSSI as

$$d = 10^{\frac{A-RSSI}{10n}}. \tag{15}$$

We use Trilateration to estimate the position of the mobile node $(x_0, y_0)$ based on estimated distances $d_i$ between mobile node and of at least three anchor nodes $(x_i, y_i)$.

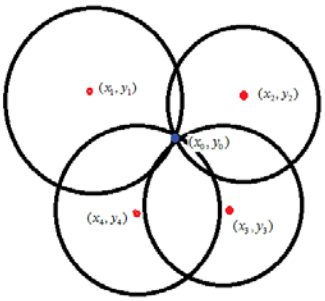



Fig. 1: Intuitively, the trilateration approach involves drawing circles with the anchor nodes as the center; for each anchor node, the distance estimated using RSSI is the radius of the circle. The mobile node lies at the intersection of these circles.

The actual distance between the mobile node and anchor node is given by

$$\begin{aligned} r_1 &= \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \\ r_2 &= \sqrt{(x_0 - x_2)^2 + (y_0 - y_2)^2} \\ r_3 &= \sqrt{(x_0 - x_3)^2 + (y_0 - y_3)^2} \\ r_4 &= \sqrt{(x_0 - x_4)^2 + (y_0 - y_4)^2} \end{aligned} \tag{16}$$

The error $e_i$, between the estimated distance,$d_i$, and the actual distance, $r_i$ is given by

$$e_i = d_i - \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}. \tag{17}$$

Ideally we would like to minimize this error

$$e_i = d_i - \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} = 0. \quad (18)$$

Simplifying, we get

$$2x_0(x_k - x_1) + 2y_0(y_k - y_1) = d_i^2 - d_k^2 - x_i^2 - y_i^2 + x_k^2 + y_k^2. \quad (19)$$

This equation is linear and can be represented as

$$Ax = B$$

where

$$A = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) \\ 2(x_k - x_2) & 2(y_k - y_2) \\ . & \\ . & \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1})] \end{bmatrix} \quad (20)$$

$$B = \begin{bmatrix} d_1^2 - d_k^2 - x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ d_2^2 - d_k^2 - x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ . \\ . \\ d_{k-1}^2 - d_k^2 - x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix} \quad (21)$$

and

$$x = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

The least squares solution for $Ax = B$, is given by

$$x = (A^T A)^{-1} A^T B$$

Hence, the estimated position of the mobile node can be computed from above equation.

### 3.3 *Fusing the Position Estimates*

The approach to measurement fusion used in our model is to weight the individual measurements from each sensor and then track those fused measurements by a Kalman Filter to obtain an estimate of the state vector. Since the measurement noise is independent for accelerometer sensor and RSSI, the equation for fusing the measurement vectors ${Z_k}^1$ and ${Z_k}^2$, for minimum square estimate is given by

$$\mathbf{Z}_k = \mathbf{Z}_k^1 + \mathbf{R}_k^1(\mathbf{R}_k^1 + \mathbf{R}_k^2)^{-1}(\mathbf{Z}_k^2 - \mathbf{Z}_k^1), \quad (22)$$

where $\mathbf{R}_k^1$ and $\mathbf{R}_k^2$ are the covariance matrices of the measurement vector of sensor 1 and sensor 2. The measurement noise covariance matrix of fused measurement $\mathbf{Z}_k$ is derived by

$$\mathbf{R}_k = [(\mathbf{R}_k^1)^{-1} + (\mathbf{R}_k^2)^{-1}]^{-1}. \quad (23)$$

The fused estimate measurements are then used to estimate the state vector $\mathbf{X}_{k|k}$ by applying extended Kalman Filter equations.

## 4 RESULTS

In this section, we present results from our experiments to estimate the position of the mobile nodes.

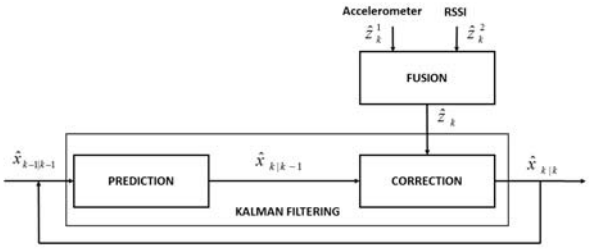



Fig. 2: Acceleration and RSSI data are fused to estimate the position of the mobile node.

### 4.1 *Experimental Setup*

Experiments were carried out in an indoor environment to estimate the accuracy of the position estimates. All the antennas on the motes were fixed at an angle of 90 degrees at the mounting surface. The average RSSI values from the mobile node were collected at a base station.

The physical dimension of the test-bed is 9m x 10m where a total of 4 anchor nodes were placed at four corners of the room. The experiment was conducted when interference from human activities was minimal. Before the experiment was conducted, several preparatory steps were performed. First, the mobile node was rotated slowly in all orientations to examine the orientation effect. After that, the signal propagation constant was calculated for each anchor node in this experiment environment. During the experiment, mobile node was manually moved at steps of 0.5m. The accuracy of the system was found by comparing the position estimated by this system with the predicted position.

### 4.2 *Parameter Determination*

Values for $n$ and $A$ were determined as described in the preceding section. To model the propagation characteristics of the receiver antennas, an off-line calibration phase was conducted. 100 RSSI measurements were taken at each place in 0.5m increments after placing the mobile node at various distances from the anchor nodes. The resulting model parameters are represented in Table I. Calibration results based on several static locations reveal the drawback of uniform computation as propagation constants are different for different anchor nodes in RSSI-Distance conversion formula. The variation of signals is due to attenuation of the signal on the medium surrounding the reference point.

### 4.3 *Localization Accuracy*

The mobile node and the anchor nodes were programmed to exchange data packets; from each message, RSSI, LQI, Acceleration values and node ID were extracted. The Average, Maximum and Minimum values was calculated for each anchor node and results were sent to the base station.

TABLE I: Path Loss Constants

| Node ID | $A$ | $n$ |
|---|---|---|
| 12 | -46.86 | 2.45 |
| 13 | -54.02 | 1.64 |
| 14 | -45.51 | 2.74 |
| 15 | -44.70 | 3.13 |

The ids of the anchor nodes are shown in the first column; for each anchor node, the transmitted power (in dBm) at a distance of 1m and the path loss constant obtained from calibration are shown.

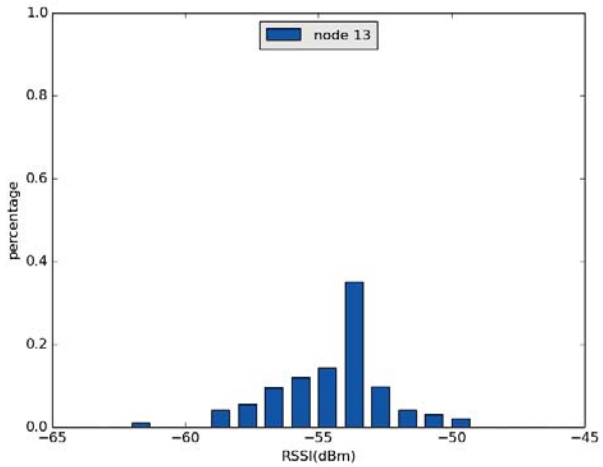



Fig. 3: RSSI values measured during calibration confirmed that the Extended Kalman Filter could be used because the distribution is Gaussian.

The average RSSI of all anchor nodes vs Distance between the anchor and mobile node in indoor environment is shown in the Figure 4. We observe that RSSI decreases with increasing distance and has larger variation due to effect of fading and shadowing. Due to the presence of walls each of the signal reflected takes a different path. So even a slight change in node's position resulted in a significant difference in received power.

Figure 5 depicts the distance traveled by the mobile node as obtained from the results of the accelerometer. The graph represents the acceleration along $x$-axis where position and velocity estimates along $x$-axis are obtained by double and single integration of the acceleration values respectively. Vibrations were kept minimum by performing the experiment on a smooth surface. The position estimation with information from accelerometer is susceptible to errors because of integration errors.

After calculating the estimated distances, these distances were compared with actual ones so that the errors can be evaluated.Figure 6 shows a comparison of fusion and no fusion distance with predicted distance for reference node 14.The average variation of the fusion filtered distance from the actual distance is reduced, which implies that the accuracy is improved compared with individual sensor position estimation.

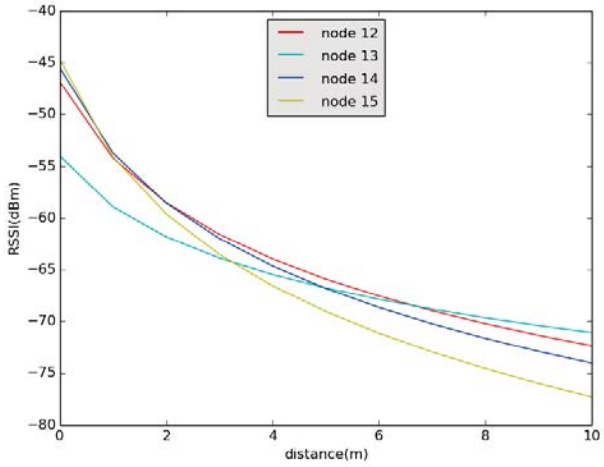



Fig. 4: The measured RSSI values confirmed that there is a non-linear relationship with distance. The relative location of the nodes significantly affect the measured value.

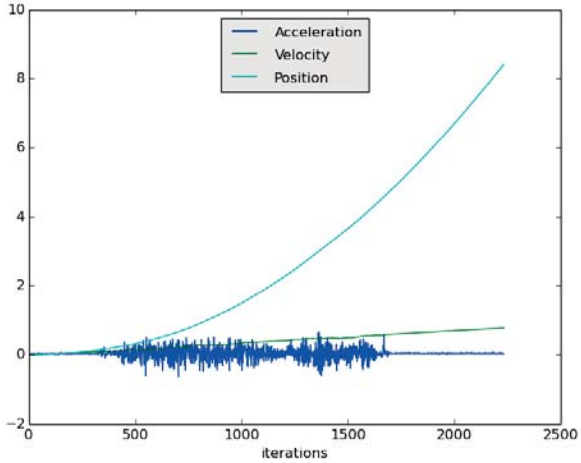



Fig. 5: The noise in the accelerometer data are a likely contributor to the error in the computed velocity and position.

Figure 7 shows the co-ordinates obtained by trilateration. The average localization error without fusing data from the two sources was 2.885m; this error error was reduced to 0.62m after fusion.

## 5 Conclusions

We presented an approach to estimate the position of a mobile node in an indoor environment that used an Extended Kalman Filter. The position estimate was obtained using RSSI values collected by exchanging beacon messages with anchor nodes, and accelerometer data sensed on the mobile node. We compared the accuracy of the position estimate that was obtained without fusing data from the two sources and by fusing the data based on experimental results. The results showed that fusing the estimates obtained using RSSI and acceleration data reduced the absolute position error from

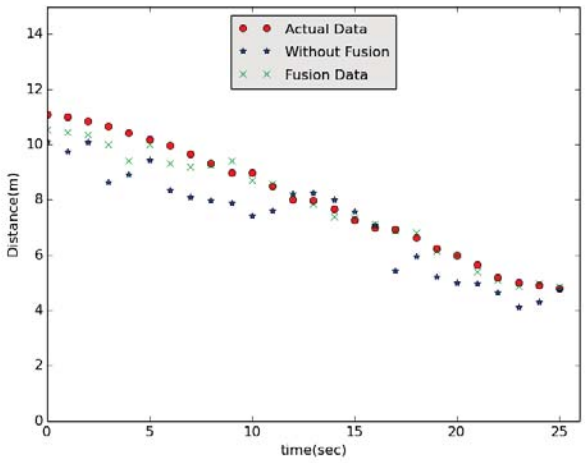
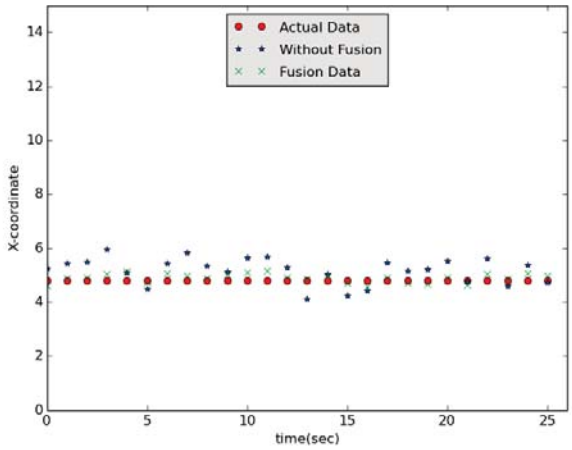


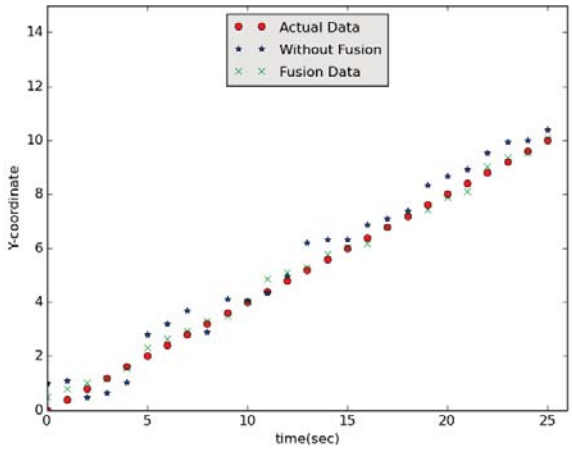

Fig. 6: Fusing the data from the accelerometer with the RSSI data helps to reduce the variation in the distance estimate with respect to the actual position.

2.88m to 0.62m. Thus, we can conclude that fusing data from multiple sources is indeed a viable strategy to improve the position estimates.

The experimental activities revealed that a large number parameters must be manipulated to achieve high-resolution position estimates using such an approach. In particular, the orientation of the nodes, the transmission power used, calibration and other environmental effects affect the fidelity of the estimates. If the transmission power is too high, the differences in RSSI values are not significant. Several experiments were conducted to derive the value of $n$ and $A$ and this value can be different for each anchor node. In the future, we are interested to carry out a more comprehensive set of experiments with a larger number of anchor nodes. In addition, we would also seek to understand how fusing multiple inertial sensors affects the accuracy of the position estimates.




(a) X-co-ordinate of the mobile node obtained by trilateration with and without fusion.




(b) Y-co-ordinate of the mobile node obtained by trilateration with and without fusion.

Fig. 7: It is observed from the figure that by fusing accelerometer and RSSI data and by applying EKF the error due to RSSI fluctuations and accelerometer drift is minimized.

## REFERENCES


[1] K.Laith and J.Peter, "Scaled unscented kalman filter for rssi-based indoor positioning and tracking," *Next Generation Mobile Applications, Services and Technologies*, pp. 132 – 137, 2015.

[2] W.Dalei, C.Dimitris, Y.Kamal, and B.Rached, "Node localization in robotic sensor networks for pipeline inspection," *IEEE Transactions on Industrial Informatics*, vol. 12, pp. 809 – 819, 2015.

[3] L.Xiaowei, M.Farah, and S.Hichem, "Decentralized localization using radio-fingerprints and accelerometer in wsns," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, pp. 242 – 257, 2015.

[4] G.Welch and G.Bishop, "An Introduction to the Kalman Filter," in *Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques*, 2001, pp. 1–41.

[5] H. Dulimarta and A. Jain, "Mobile robot localization in indoor environment," *Pattern Recognition*, vol. 30, pp. 99–111, 1997.

[6] E. Baumgartner and J. Yoder, "Enhancing mobile robot location estimates using sensor fusion," *Dynamic Systems and Control Division*, pp. 137–144, 1995.

[7] Y. Bar-Shalom, X. R. Li, and T. Kirubajan, *Estimation with applications to tracking and navigation*. Editions Wiley-Interscience, 2001.

[8] D.Lili and Z.Kai, "Sensing and control of mems accelerometers using kalman filter," *24th Chinese Control and Decision Conference (CCDC)*, pp. 3074 – 3079, 2012.

[9] L. F. J. Borenstein, H.R. Everett and D. Wehe, "Mobile robot positioning: sensors and techniques," *Robotic Systems*, vol. 14, pp. 231–249, 1997.

[10] Q.Gan and C.J.Harris, "Comparision of Two Measurement Fusion Methods for Kalman Filter Based Multisensor Data Fusion," *Aerospace and Electronic Systems*, vol. 37, pp. 273–279, 2002.

[11] K.C.Change, Z.Tian, and R.K.Saha, "Performance evaluation of track fusion with information filter," *International Conference on MUltisource-Multisensor Fusion*, pp. 648–655, 1998.

[12] A. S.Paul and E. A.Wan, "Rssi-based indoor localization and tracking using sigma-point kalman smoothers," *IEEE Selected Topics in Signal processing*, 2009.

[13] W.Chai, C.Chen, E.Edwan, J.Zhang, and O.Loffeld, "Ins/wifi based indoor navigation using adaptive kalman filtering and vehicle constraints," *Positioning Navigation and Communication*, 2012.

[14] A. Colombo, D. Fontanelli, D. Macii, and L. Palopoli, "Flexible indoor localization and tracking based on a wearable platform and sensor data fusion," *IEEE Transactions on Instrumentation and Measurement*, 2014.

[15] *LIS344AL - MEMS Inertial Sensor 3-Axis Ultracompact Linear Accelerometer*, ST, 12 2007.

# Using Simplified Grammar for Voice Commands to Decrease Driver Distraction

**Ashraf Gaffar** [1], **Shokoufeh Monjezi Kouchak**[2]
[1]Ira.A Fulton School of Engineering, Arizona State University, Arizona, USA, agaffar@asu.edu
[2]Ira.A Fulton School of Engineering, Arizona State University, Arizona, USA,smonjezi@asu.edu

**Abstract -** *Today's car is considered a Cyber Physical System implying a shift of control and management of the car's features to complex onboard software. Since the computer has taken control over most of the in-car driving and entertainment (infotainment) activities, driver distraction has become one of the primary reasons for car-related fatal accidents. As a result, we need to decrease driver distraction despite the growing presence of cockpit interactive digital technologies. Using voice-based commands is a potential solution to decrease visual distraction but it causes significant cognitive distraction. In this paper, we present a new method for voice-based commands to decreasing the cognitive distraction while using command based voice inputs.*

*Keywords:* ***distraction; human car distraction; cognitive load, cyber-physical systems***

## 1 Introduction

Driver distraction is one of the main reasons of car-related fatal crashes. NHTSA estimates that around 25% of traffic crashes in United States of America are caused by driver distraction [1]. Driver distraction is generally agreed upon to be caused by every person, event or object that persuades the driver to shift her attention from the fundamental task of driving [2]. Although distraction is not directly or readily observable, it is clearly manifested in why -under some conditions- the performance of a task is degraded [3]. These conditions are typically considered related to a distractor.

As mentioned earlier, today's car system is a Cyber Physical System [4]. The car lifecycle right from its early stage of design and build phase to its running phase is controlled by software. Automotive industry is a software-heavy domain with software dominating the design, manufacturing, and maintenance of a car. Besides, the car itself is loaded with complex hardware and software components. Considering this, we focus our work on the interactive relationship between a driver and the digital environment of the cockpit, mostly in the infotainment system built in in the mid console, also known as the center stack. The modern infotainment system has three main groups of interactive software:

1. **Built-In Features:** Standard car information and local entertainment group originally loaded with the car or downloaded by the driver.
2. **Bluetooth Connectivity**: Providing additional features accessible via externally connected smart devices. These typically have hundreds of additional apps, contact lists, personal calendars, and other features. While originally designed for non-driving smart device user, they can often be synchronized, displayed and managed from the infotainment screen
3. **Web Connectivity**: Onboard access to the World Wide Web, which can add a virtually unlimited number of interactive online features with a potential for serious distraction.

The main impact on the negative side of advanced Infotainment system is growing driver distraction by allowing drivers' access to such apps and features originally not intended for driving environment. These can add to the large number of other distracting tasks like mobile phone use or talking to other passengers [5].

One challenge of measuring driver distraction is the fact that there are mixed types of distraction, and they can affect driver in different ways. The research community identifies four main types of driver distraction that can be present in isolation or together: *Visual, auditory, manual and cognitive* distraction [2]. *Visual* distraction occurs when driver focuses on entertainment system instead of keeping eyes on the road. *Auditory* distraction occurs when the sound in the car such as passenger conversation, traffic noise or music masks the warning noise that is necessary for self-monitored driving performance. *Manual* distraction happens when the driver

takes hands off the wheel to do non-driving related activity like eating or drinking. *Cognitive* distraction happens when the driver is absent-minded (looking but not seeing) or when the driver is busy with deep thinking about something [2][6].

A major research challenge is that depending on the method of study, evaluation, and interpretation, similar activities can be considered highly positive and highly negative in different scientific studies. For example, listening to music has been considered as harmless activity during driving in some papers [7] but was considered a distraction in some other papers [1]. Few of the previous research works say that listening to music during driving has effect on driving performance in two ways: distraction and mood effect.

Listening to music causes audio, visual and biomechanical distraction. According to a research in the United States of America 91% of drivers manipulate their cars' audio system control and it takes 1% of driving time. They adjust their audio systems 8 times every hour and each interaction took 5.5 seconds.

Analysis of police accident report provided an evidence of a connection between road accidents and using in-car infotainment system, such as audio system. This analysis showed that in-car infotainment system was the first as well as the second reason for in-car distraction. [1]

Texting by mobile phone or using iPad can cause all 4 types of distractions. So using them during driving increases the driving errors significantly [8] and [9]. For example, using cell phone during driving has negative effect on lane keeping and event detection [10]. People who use phone during driving are four times more likely to meet with an accident that will result in hospital attendance; Age group and sex don't influence the risk of accident when driver uses mobile phone during driving. Even using the hands-free feature is not helpful and it doesn't decrease the chance of accident [11].

Although the tasks that don't need visual attention or manual interaction cause less distraction, the potential for cognitive distraction has to be considered [12]. It is hard to determine attention status of driver so it is hard to solve cognitive distraction problem, because there isn't any one-to-one relation between car and brain activities, so some ambiguity will be in the consequence of estimate [13].

The number of applications on the infotainment system in modern cars is expected to increase significantly during the next few years [14], so finding new methods to reduce the driver distraction that is caused by infotainment system will reduce the fatal crashes significantly. In this paper, we explain a new method to decrease the distraction that voice based commands of infotainment systems causes.

Earlier researches have proven that using the voice-based commands in the car user Interface (UI) reduces driver distraction for two reasons:

1. It reduces the visual distraction because the driver doesn't need to take his eyes off the road and search among lots of apps to find his favorite app. Using touch screen instead of voice commands increases visual distraction significantly since the driver needs to several steps navigation for finding each application.

2. It reduces manual distraction since the drivers doesn't need to take his hands off the steering wheel when he is using the voice commands.

Although using voice commands reduces both visual and manual distraction, it causes cognitive distraction and audio distraction.

Our hypothesis is that much of the cognitive load associated with voice command is related to the system not being able to understand commands all the time, forcing users to repeat or change the way they issue their command, taking a significant portion of their attention off the road trying to perceive the system ill-response and trying to figure out a different way to communicate their desire. We have used simplified voice based commands instead of all-natural voice based commands with complicated grammar. It was found that using simplified grammar significantly decreases distraction. In the following sections, we explain our hypothesis followed by the description and type of participants who took part in the experiment and the equipment that we used in the experiment. The results of the experiment are given in the form of statistical analysis.

# 2 Voice commands experiment

## 2.1 Using simplified grammar

Although voice based commands decreases the visual and manual distraction, cognitive distraction and audio distraction can significantly increase and even reach a dangerous level. Using complicated grammar increases the cognitive distraction since the drivers makes more mistakes such as mispronunciation and misunderstanding, the chance of misdiagnosed voice command increases so the driver needs to repeat each voice commands several times as a result the cognitive distraction increases. When the driver uses voice commands with simple grammar the number of his mistakes decreases so the number of misdiagnosed voice commands decreases. Less misdiagnosed voice commands means less cognitive distraction. Listening to sentences during driving causes distraction so reducing the number of misdiagnosed voice commands by using voice commands with simple grammar reduces the audio distraction too.

We designed an experiment to find out the different between response time and the number of errors when the driver uses voice commands with simple grammar and when the driver uses the voice commands with complicated grammar. Response time has direct correlation with cognitive distraction so reducing the response time would decrease the cognitive distraction. Less response time for a command shows that this command causes less cognitive load so it reduces the driver distraction.

Does using simple grammar in speech mode communication instead of sophisticated voice commands reduce response time? Our experiment's results show that using simplified grammar for the voice commands in speech based interaction reduces response time and distraction level compared to the voice commands with complicated grammar.

## 2.2 Equipment and participants in experiment

The experiment was conducted in a Drive Safety Research simulator DS-600(Figure 1). The DS-600 is fully integrated, high performance, high fidelity driving simulation system which includes multi-channel audio/visual system, a minimum 180° wraparound display, full width automobile cab (Ford Focus) including windshield, driver and passenger seats, center console, dash and instrumentation and real time vehicle motion simulation.

Figure 1:Hyper Simulator in the Experiment

It renders visual imaginary at 60 frames per seconds on a sophisticated, out-the-window visual display with horizontal field-of-view. It also includes three independently configurable review mirrors.

An Android application was developed to display the user interface with numbers ranging from 1 to 9. The application was hosted on the Android v4.4.2 based Samsung Galaxy Tab 4 8.0 that was connected to the Hyper Drive Simulator.

We used 45 volunteers in our experiments each belonging to different sector participle in our experiment. We asked them to drive on a previously programmed route. During driving they were asked to interact with both voice based system with simplified grammar and normal grammar.

In this experiment, normal grammar User Interface (UI) referred to usual commands present in cars that have speech based communication models to interact with the center stack and simplistic Grammar user interface, a user interface that has voice commands with simple grammar, used in order to perform a set of operations during driving. Figure 2 and Figure 3 show an example of 2 categories of commands.

We measured two variables in this experiment: Response Time (in milliseconds) and Errors.

Response time: Response time is the total time taken by a person to interact with the system using voice command. Resulting in decreased response time causing less distraction as the driver comparatively spends less time on the secondary task.

Number of errors: The count of errors refers to the number of times that the driver changed the lanes incorrectly and hit other cars or buildings while using the voice commands. More errors imply more distraction.

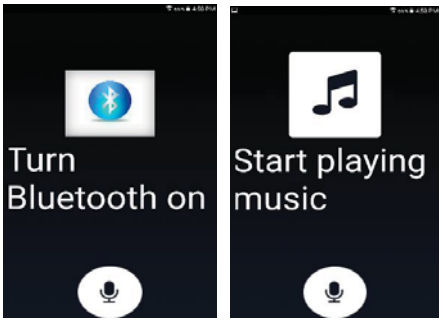



Figure 2:Normal command

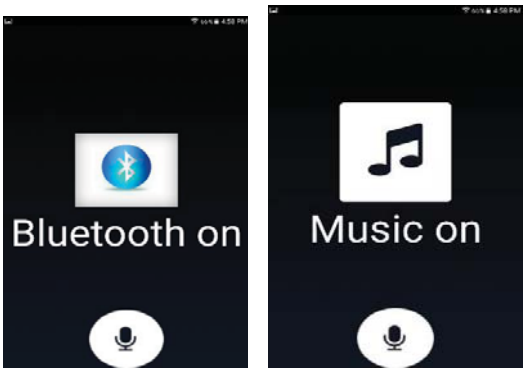



Figure 3:simple command

We used two statistical methods to analysis our experiment. The first statistical method that was used to analysis this experiment is "comparing two alternatives".
Comparing between two alternatives used to compute the mean of differences of the paired measurements. The confidence interval C1 and C2 are determined with a confidence level, say 95%. This statistical method helped us

to determine if the results of our experiment are statistically significant.

$$c_1 = \bar{d} - z_{1-\alpha/2}\frac{s_d}{\sqrt{n}}$$

$$c_2 = \bar{d} + z_{1-\alpha/2}\frac{s_d}{\sqrt{n}} \quad (1)$$

If c1<0 and c2 > 0, then the measurements are statistically insignificant. If c1 and c2 >0 or c1 and c2 < 0, then we can conclude the measurements are statistically significant.

The second statistically method that was used to analysis this experiment is analysis of variance (ANOVA). Analysis of variance is a technique to divide the total variation observed in an experiment into different meaningful components. This technique assumes that the errors in the measurements for different setting are independent with normal distribution.

By using ANOVA we can compute the variation that observed because of changing the alternatives and the variation that observed because of the error in measurement. The measurements are significant when the variation that observed because of changing alternatives is more than variation that observed because of the errors in measurement. The variation due to alternatives SSA and variation due to errors SSE are calculated using the below equations. Where k being the number of alternatives. In our case k = 2, and n is the number users in the experiment. n = 20.

$$SST = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}(Yij \cdot Yij)}{kn} - \frac{(Yij \cdot Yij)}{kn} \quad (2)$$

$$SSA = \frac{\sum_{j=1}^{k}(\sum_{i=1}^{n}(Yij \cdot Yij))2}{n} - \frac{(Yij \cdot Yij)}{kn} \quad (3)$$

$$SSE = SSA - SST \quad (4)$$

The estimate of variance of alternatives and errors are calculated using:

$$S2a = \frac{SSA}{k-1} \qquad S2e = \frac{SSE}{k(n-1)} \quad (5)$$

F value is computed using:

$$F = S2a / S2e \quad (6)$$

This calculated F value is compared with the value $F_{[1-\alpha;(k-1),k(n-1)]}$ obtained from the table of critical F values. Then we can say the variation due to alternatives is greater than variation due to errors with a confidence level of (1 -α).

We will use this statistical analysis to prove that the variation of response time is significantly due to the change in alternative (changing the UI setting of Center stack) rather than the variation due the different users. This also gives a statistical estimate of the effect on the response time due to change in alternative and change in user.

## 2.3 Procedure

Before the start of the experiment, each volunteer was given few minutes to get acquainted with the simulator. After that, they were asked to drive two times. First time, they were asked to do secondary tasks like operating the center stack by giving voice commands with complicated grammar. The response time and errors of all volunteers were recorded.

In the second time, they were asked to interact with center stack, during driving, by giving different set of simple commands (commands with simple grammar) and response time and errors were recorded again. This operation was repeated at three locations:

1. Very fast left when the cars would be coming from different direction.
2. When the cars suddenly change on the freeway.
3. On the left turn of the road when they encounter pedestrians crossing.

Figure 4 shows the response time for each volunteer in milliseconds when they operated on commands with simple and complex grammar.

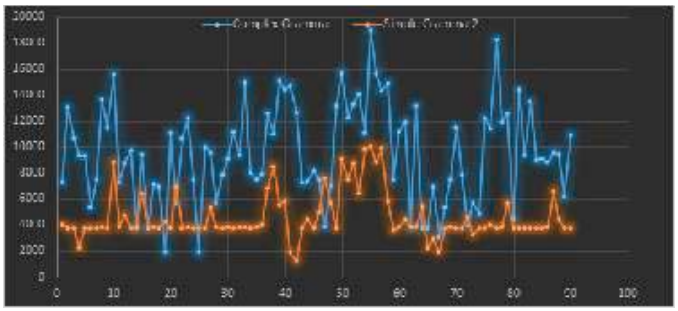


Figure 4:Response time in milliseconds

Figure 5 shows the number of errors for each volunteer when they operated on commands with simple and complex grammar.

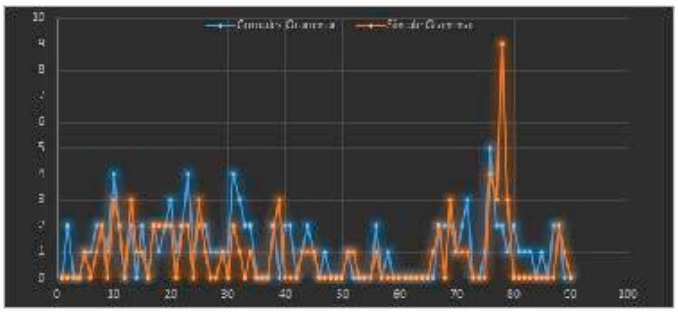


Figure 5:Number of errors for each volunteer

Table1 shows the average and standard deviations in the response time of our experiment. Response time 1 is the Response time for commands with complex grammar and Response time 2 is the response time for commands with simple grammar:

Table 1

| | Response time 1 | Response time 2 | Difference |
|---|---|---|---|
| **Mean** | 9223.76 | 4229.6 | 4994.16 |
| **Standard Deviation** | 3372.03 | 1416.63 | 3428.51 |

## 2.4 Results

CI (Confidence Interval) is a combination of sample size and variability. In order to interpret the CI for population mean, we have to assume that samples were chosen randomly and independently from a population and that the values were distributed according to Gaussian distribution. When we accept this assumption, there is 95% chance that 95% of CI contains the true population.

We found the C1 and C2 for both response time and error. You can see the results in Table 2.

Table 2

| | C 1 | C 2 |
|---|---|---|
| **Response time** | 4276.06 | 5712.25 |
| **Error** | 0.254381577 | 0.67851756 |

Since C1 and C2 for response time are positive, we can say that the measurements obtained are statistically significant and the response time of alternative2 (voice command with simple grammar) is less than alternative1 (voice command with complex grammar), so with 95% of confidence we can say that using voice commands with simple grammar is better than using voice commands with complex grammar because they reduce the response time.

Similarly For the errors that was measured in this experiment according Table 2 since c1 and c2 for errors are positive, we can say that the measurements obtained are statistically significant and the number of errors with alternative 2 (voice commands with simple grammar) is less than alternative 1(voice commands with complex grammar), so with 95% of confidence we can say that using voice commands with simple grammar is better than using voice commands with complex grammar because they reduce the number of errors.

In this experiment we used ANOVA in order to analysis the collected data statistically. Table 3 shows the results of using ANOVA to statistical analysis of volunteer drivers' response time in this experiment. According to this table SSA/SST=0.865 thus we can say 86.5% of total variation is due to difference between two alternatives options. And SSE/SST=0.134 so we can say that 13.5% of total variation is due to noise in the measurements of different users.

In this table F-computed is greater than F-tabulated so at 95% confidence we can say that the differences seen in the response time by changing the alternatives is statistically significant.

Table 3

| | |
|---|---|
| **SST** | 11665942960 |
| **SSA** | 10094439615 |
| **SSE** | 1571503346 |
| **SSA/SST** | 0.865291357 |
| **SSE/SST** | 0.134708643 |
| **F-computed** | 1143.3703 |
| **F-tabulated** | 3.087 |

Table 4 shows the results of using ANOVA to statistical analysis of number of errors of drivers in this experiment. According to this table SSA/SST=0.397 thus we can say 39.7% of total variation is due to difference between two alternatives options. And SSE/SST=0.602 so we can say that 60.02% of total variation is due to noise in the measurements of different user. Since F-computed is greater than F-tabulated, at 95% confidence we can say that the differences seen in the response time by changing the alternatives is statistically significant.

According these statistical analyses we can say that using voice commands with simple grammar decreases the cognitive distraction since it reduce the number of errors also it reduces the response time. Commands with less response time cause less cognitive distraction because the driver spends less time for secondary task during driving. In addition in this experiment the number of errors of drivers reduces when they use voice commands with simple grammar. It shows that they could focus better on their primary task (driving) when they use voice commands with simple grammar so they made fewer errors.

Table 4

| | |
|---|---|
| **SST** | 456.994506 |
| **SSA** | 181.4722531 |
| **SSE** | 275.5222529 |
| **SSA/SST** | 0.39709942 |
| **SSE/SST** | 0.60290058 |
| **F-computed** | 117.2393907 |
| **F-tabulated** | 3.087 |

## 3 Conclusions

Using voice commands in car user interface can reduce visual and manual distraction but using this kind of commands causes other kinds of distraction. It causes cognitive distraction and audio distraction. Misdiagnosed voice commands cause cognitive distraction because the user should repeat a voice command several times and sometimes the driver should divert to another kinds of command. Sometimes the noise of environment causes misdiagnosed. By considering this fact that speech recognition systems are not ideal even with the best technology in order to improve the quality of voice based commands and decreasing the response time of driver and decreasing the number of driver's error we should use voice commands with simple grammar because this kinds of commands are easy to remember and easy to pronounce so the response time decreases if we use the voice commands with simple grammar. In this paper we talk about the results of our experiment that compared the response time and number of errors in two scenarios. In the first scenario we used commands with complex grammar and in the second scenario we used commands with simple grammar. We observed that the response time reduced by half with simplified grammar. Also the errors reduced by 1/3rd when we used simple grammar for voice based commands. Driver was taking more attempts to get the command right when using complex grammar and he was getting the command right at the first attempt when commands had simple grammar. So we can say that using commands with simplified grammar decreases driver distraction when interacting through speech commands in car.

## 4 References


[1] http://www.nhtsa.dot.gov/announce/testimony/distractiontestimony.html, March 2nd

[2] Nicola Dibben and Victoria J.Williamson. " An exploratory survey of in-vehicle music listening", Sychology of music, Vol No.35(4):571-589, pp.571-576, 2007.

[3] Dean Sugano. "cell phone use and motor vehicle collisions a review study", Legislative reference Bureua sate capital Honolulu Hawaii, report no.4,2005.

[4] https://en.wikipedia.org/wiki/Cyber-physical_system,March 5h

[5] William P Berg and Dirk J Dessecer ,"Evidence of uncouncious motor adaption to cognitive and auditory distraction",Adaptive behavior,Vol No. 21(5) 346-355,pp.347-355,2013.

[6] Michael Bull."Automobility and power of sound" ,Theory culture and society, vol No. 4-5 243-259,pp.244-258,2004.

[7] William Consiglio,Peter Driscoll, Matthew Witte, and William P Berg. "Effect of cellular telephone coversations and other potential interface on reaction time in a braking response" , accident analysys and prevention,VOL No.35,465-500, pp.496-500,2002.

[8] Mustapha Mouloua, Amber Ahren and Edward Rinalducci ."the effect of text messaging on Driver Distraction:A bio-behavioral analysys",Processing of the human factors and ergonomics society 54th annual meeting, Vol No. 19 1541-1545, pp.1541-1545,2010.

[9] Mustapha Mouloua, Daniela Jaramillo and Janan Smither ."the effect of ipod use on Driver Distraction",Processing of the human factors and ergonomics society 55th annual meeting,pp.1583-1586,2011.

[10] Jeff K.Caird and Chip T.Scialfa."A meta-analysis of driving performance and crash risk associated with the use of cellular telephone while driving", proceeding of the third international driving symposium on human factors in driver assessment,training and vehicle designe,pp.478-485, 2005.

[11] Suzzane P.McEvoy, Mark R.Sterenson, Anne T.McCartt, Mark Woodword, Clair Haworth, Peter Palamara and Rina Cercarella, "Role of mobile phones In motoe vehicle crashes resulting in hospital attendance : acase study" British Medical journal, vol.331,No. 7514, Aug .20-27, pp.428-430, 2005.

[12] Joanne L. Harbluck , Y Ian Noy, Patricia L.Trbovial and Moshe Eizenman. "An on-road assessment of cognitive distraction:Impacts on drivers visual behavior and braking performance.", Vol No. 39(2),pp.372-379,2006.

[13] Nicola Dibben and Victoria J.Williamson." An exploratory survey of in-vehicle music listening", Sychology of music,Vol No.35(4), pp.571-589, 2007.

[14] www.speechtechmag.com, March 5th

# Wireless Hardware-in-the-loop Simulations of an Autonomous Electric Vehicle with an Interoperable Real-time Simulation (iRTS) Platform

Ankurkumar Patel, Fnu Qinggele, and Yong-Kyu Jung
{patel070, qinggele001, and jung002}@gannon.edu
Electrical and Computer Engineering, Gannon University, PA, USA

**Abstract**
*A real-time simulator (RTS) is a vital tool for rapidly designing and testing real-time embedded applications including an autonomous electric vehicle (AEV). A wirelessly inter-operable hardware-in-the-loop (HIL) real-time simulation (iRTS) platform for researching an AEV system was developed. The microkernel-based multi-core RTS is interfaced with/without wire (IEEE 802.3u/802.11g/ 802.15.4) to consoles and integrated to an FPGA-based extension module for accurate and high-speed HIL simulations of an AEV prototype. The proposed iRTS platform provides swiftly handling different complexity and scale of the AEV testing while satisfying real-time constraints (i.e., 7 μs resolutions and 0.3% HIL simulation error) of the AEV models and prototypes.*

## I. Introduction

The increasing demand for real-time simulators (RTS) in the automotive industry significantly grows the number of RTSs used even in academia [1]. Specifically, hardware-in-the-loop (HIL) simulation becomes vital for the automotive electronics systems that required comprehensive evaluations under more realistic conditions with real-time constraints. The HIL simulation [2] offers fast cost-effective verifications and eludes the jeopardy of incidents for hundreds of microcontrollers employed in the electric vehicle (EV) electronics. Such HIL simulations are usually initiated after series of software-in-the-loop (SIL) simulations without any prototypes of the EV components. In order to avoid delays on an EV development process between the different SIL simulations and the realistic HIL simulations with the autonomously operating prototypes in an EV in different levels of development stages, a RTS must be capable of interfacing to and interoperating with hundreds of autonomously operating models and prototypes under the sufficiently flexible environment. Thus, capability of a wireless RTS for the diverse HIL simulations contributes in the development of electronic components of an autonomous electric vehicle (AEV). Developers swiftly test the prototypes at the different levels of development and discover a breakthrough with a series of the interoperable HIL simulations. The proposed interoperable RTS (iRTS) platform has been developed in order to successfully utilize the iRTS in AEV electronics research.

Various embedded components and systems have been employed into transportation systems, which still need to provide more systematic assistance to operators of the driving systems than current human-initiated systems. For instance, a drowsy driver must be alarmed by a safety subsystem installed in the vehicle. Furthermore, the current and next generations of the autonomous vehicles must be well equipped with various sensor systems [6] integrated with microcontrollers in order to lead the tendency for the future autonomous transportation systems. Accordingly, an autonomous driving model for real-time simulation has been studied. The operations of an autonomous driving prototype were controlled by the routing path selection (RPS) algorithm [3]. There are various similar algorithms [4]. The presented research work was applied for several aspects of automated-assistance, including driving circumstance, such as weather and traffic, road condition and structural features, and driver's condition, including driving time and distance. Verification of both software and hardware components and an integrated system is another important aspect in succeeding in the proposed research. Therefore, the iRTS was developed for evaluating the proposed concept and for extending the capability of the existing RTS. The extended iRTS is expected to offer that of a real prototype capable of running outside of the iRTS while running a few virtual prototypes on FPGA and executing software models concurrently in the iRTS.

Section 2 reviews applications and features of real-time simulations. Section 3 describes the architecture and operation of the iRTS. Section 4 expresses evaluation of the iRTS integrated to an EV model and prototype via a wireless interface for the rapidly extensible HIL simulations. The evaluation results and analysis of the iRTS for the AEV are also described in Section 4. Section 5 concludes the proposed work.

## II. Related Work

An efficient and perceptive simulation environment is required to allow for quickly discovering and evaluating alternative design and control strategies [5]. In particular, an efficient management including model integration, input/output (I/O) allocation, accurate execution, and systematic test creation is one of the key parts for the successful utilization of the RTS. Professional RTSs have been utilized in order to provide adequate interfaces

ISBN: 1-60132-433-2, CSREA Press ©

for various aspects of software and hardware models, I/O allocations, and test analysis. Other application-specific RTSs were developed for different applications, such as the electric control unit in EV [5], fuel cell in hybrid EVs [7], electric and hydraulic systems in avionics [8], and power electronic system [9].

In general, HIL simulation becomes crucial for the systems that require comprehensive evaluations under realistic conditions with real-time constraints. The HIL simulation [10] not only offers cost-efficient, fast verifications, but also avoids or at least mitigates risks of contingency situations for microcontrollers in the systems. The HIL simulation permits models of a part of the system to be simulated in real time with the actual hardware of the remainder of the system. In order to integrate models and real hardware together, various I/O interfaces generally provide analog-to-digital and digital-to-analog conversions (ADC/DAC), voltage conversions for different digital signaling, and signal conditioning equipment. More specific HIL simulations, such as controller HIL simulation for rapid controller prototyping and power HIL simulation with transferring real power to the hardware system, are discussed in [11]. Thus, the HIL simulation capability in RTS is instrumental in the development of EV. Developers promptly test the EV platform, evaluate the control strategy, and discover a breakthrough with the HIL simulation.

As the number of RTSs has been utilized in industry, academic version of RTSs [12, 13] are expected to embrace specific features including meaningful and relevant experience without being limited by laboratory equipment, user-friendly interfaces with increasing sophistication, the flexibility for continuous expansion, and preferred cost-effectiveness. A cost-effective academic RTS with SIL/HIL capability has been developed in order to successfully utilize the RTS in academia, especially for different disciplines including Power Electronics, Electric Drives, Embedded Systems, and Communications.

## III. Development of iRTS

Unlike an EV model running on a RTS in the previous practice, the autonomously operating AEV interacts with concurrently operating functions in the presented real-time simulation environment. The iRTS environment is capable of interfacing wirelessly to the AEV. In order to enhance the autonomously operating capability of the AEV, the dynamic motion detection system detects moving objects and obstacles from its surrounding environment and communicates with the iRTS located in a remote location via a wireless network channel. Therefore, researchers and developers can exercise their algorithms programmed in either the AEV or the RTS under the numerous road conditions and surrounding driving circumstances for collecting critical information to improve the capability of autonomously driving an AEV. In addition, the HIL simulation performing on the presented iRTS can offer sufficient flexibility for further extension via an FPGA integrated in the iRTS.

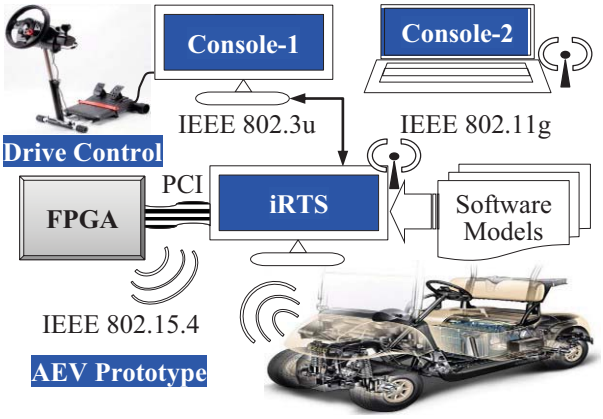



Figure 1. Organization of iRTS for the SIL/HIL Simulations with the AEV Prototype

### 3.1. Organization of iRTS

Figure 1 illustrates an organization of the iRTS. Four primary modules of the iRTS are operable for (1) modeling SIL models and analyzing simulation results with multiple consoles; (2) performing SIL/HIL simulations on RTOS (i.e., QNX 6.5) and interfacing HIL models/AEV prototype and consoles; (3) modeling HIL models with FPGA and interfacing to the AEV prototype; and (4) controlling the SIL models and the AEV prototype. In particular, iRTS is remotely accessible by multiple consoles, where SIL models are developed in Simulink/C and simulation results are received and analyzed via a 100M bps IEEE 802.3u channel and/or a 52M bps IEEE 802.11g Wi-Fi. A series of control bit streams from a driving unit is decoded in a console can be transmitted to iRTS for dynamic and remote control of the AEV prototype. This feature will be added to the current iRTS. The bit streams are already decoded. The AEV prototype can be accessible via the wireless module integrated with the FPGA module. In addition, the FPGA module provides different HIL models written in VHDL, including a PWM generator and a motor control unit, interoperating with SIL models running in the iRTS via a peripheral component interconnect (PCI) interface [16] and the AEV prototype via wireless communication channel.

### 3.2. Internal Organization of iRTS

The iRTS is a multi-core (i.e., Intel Core i7 Quad Core Processor 870 with the VT (2.93GHz, 8M)) computer executing a real-time operating system (i.e., QNX 6.5). The iRTS also contains a PCI card for the HIL simulation. Figure 2 shows an organization of the iRTS. The four cores with double thread per core are scheduled to execute the distributed tasks in parallel on the real-time operating system. The iRTS performs both SIL and HIL simulations. In particular, the iRTS provides a hardware-

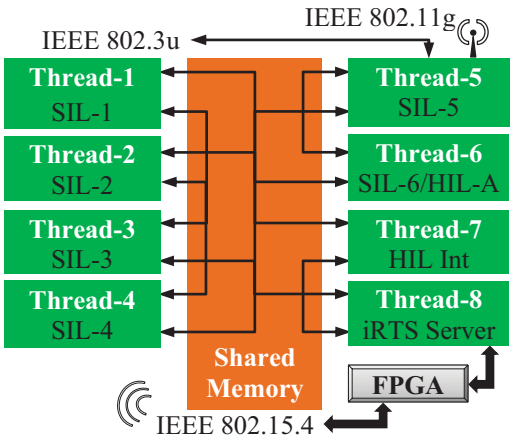



Figure 2. Internal Operations of iRTS for the SIL/HIL Simulations with the AEV Prototype: Up to eight threads (i.e., up to six SIL models and two HIL models/ interfaces) concurrently execute via a shared memory

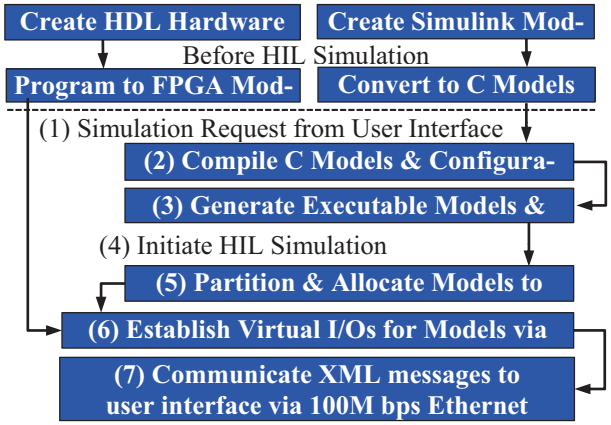



Figure 3. Operation of iRTS for the HIL Simulation

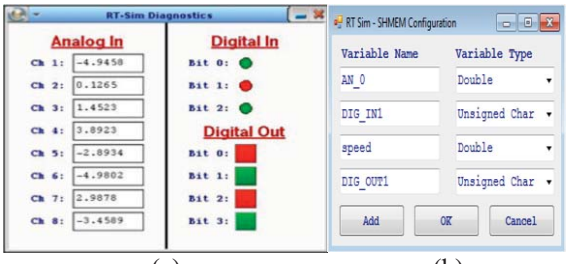
interface task to process and establish the hardware I/O for the HIL simulation via the installed PCI card which is capable of processing analog/digital I/Os. The rate of the interface task is dependent upon the rate of the fastest software model executing on the simulator. Threads 1 to 4 in core 1 and 2 are allocated for the SIL simulation. Core 3 interfaces to the wireless HIL interface and additional SIL simulation. Core 4 interfaces to the PCI for the HIL simulation and processes the socket server and communication with the console.

### 3.3. Operations of iRTS

Figure 3 illustrates the operation flow of the iRTS platform. The iRTS operates as a server, which can support a plurality of the user interface modules to initiate HIL simulations. The iRTS operation procedures are: (1) launching threads to service the user interface requests; (2) automatically compiling the C models and configurations received from the user interface module; (3) generating the executable models and configurations; (4) initiating the simulation of the models according to the configuration upon receiving a simulation start request from the user interface module; (5) generating a set of partitions to assign the different tasks to the available threads in the cores according to the partitioning scenarios built-in the iRTS; for instance, iRTS dynamically distributes a thread for the HIL simulation, another thread for managing current simulation, and the remaining thread for the SIL simulation; for instance, up to four software models run concurrently on the four threads or more than five software models can be virtually executed in parallel by sharing the same thread in different scheduled time while a remaining thread monitors core usage, dynamic task distribution and scheduling, and output management; (6) establishes virtual I/Os of concurrently operating threads; for the HIL simulation, a dedicated thread is assigned to access the PCI card for exchanging signals to the VHDL/Verilog models running on the FPGA module and/or additional HW prototypes via the hardware interface module; and (7) continuously communicates simulation outputs to the user interface module via a client-server socket-based 100 Mbps Ethernet connection as the communication layer with the socket-based architecture, which resolves the overhead of FTP (e.g., maintain the IP addresses of clients); this allows providing flexibility in any model to be executed on iRTS without any modification to the communication layer by transmitting custom XML messages to the user interface module.

### 3.4. Interface of iRTS

To deliver the output from a model to an input of the same and/or other model, the iRTS provides a dynamically configurable virtual interface, which is a flexible shared memory scheme to support seamless configuration for various real-time simulations. In addition, the contents of the shared memory are transmitted periodically to the user interface module for processing and analysis of the simulation outcomes. The iRTS also saves the same data as a precaution. Figure 4 illustrates an example of the dynamically configurable virtual interface for a HIL simulation. 'N' number of the software models connect to the shared memory region for inter-




(a) (b)

Figure 4. Dynamic Virtual Interfaces in the iRTS for HIL Simulations: (a) Analog/Digital I/O Diagnostic Tool and (b) Shared Memory Configuration Tool

facing to each other models including the external hardware models in the FPGA module and the AEV prototype via wireless connections. In particular for the HIL simulation with the external hardware, the analog and digital I/Os are interfaced to the configured shared memory locations at the specified frequency in the HIL simulation configuration. The diagnostics screen seen in Figure 4 (a) captures the interface signal values sampled by ADC and DAC in the PCI for verification of the HIL simulation interface. A diagnostic tool was configured via the configuration tool shown in Figure 4 (b) before monitoring analog and digital I/Os measured by the iRTS via the PCI.

## IV. Design of a Dynamic Motion Detection (DMD) System for an EVA

A car may be integrated with thousands of assembly parts, and the design for each part becomes more and more complicated. Many practical issues that include dynamically changing road conditions and driving circumstances are difficult to resolve via traditional methods. Sometimes, there are significant gap between the experimental results and the expected results during the field tests. In addition, the development cycle of automobile production, including unit design, trial production, local test and vehicle test, influences the entire development process. To solve these problems, real-time simulation-based prototyping becomes popular in the automotive industry [14, 15].

We have implemented the RPS algorithm to the DMD system, which enables to autonomously driving an EV prototype. Initially, the EV simulation model was developed and running on the iRTS. Then, a speed controller and other time sensitive components were transformed to parts of hardware models that integrated with the remaining simulation models of the EV. This transition permits us to identify the simulation error between SIL and HIL simulations. The HIL simulation is still limited the further evaluation of the EV prototype, including embedded software testing. Thus, an AEV prototype has been developed for test various application hardware and software that can be interoperable between components installed in the AEV prototype. Therefore, the AEV prototype provides a means to monitor behaviors of the embedded system integrated on the AEV prototype via rapidly exercising different algorithms available. In addition, wireless capability enhances flexibility and remote accessibility of the DMD system and the iRTS interface for performing the HIL simulations under the realistic test conditions.

The DMD system comprises of a wireless device installed on a movable sensor module and another wireless device installed on an FPGA-based hardware interface module [17]. Thus, the DMD system not only detects moving objects and obstacles, but also measures the distance between the objects/obstacles and the AEV prototype. The AEV prototype is capable of driving all four wheels independently. The AEV can hold multiple controllers, sensors, and radio frequency modules, and so forth. An ultrasound sensor module or an image sensor was installed on the AEV prototype for testing and developing algorithms related to the DMD system.

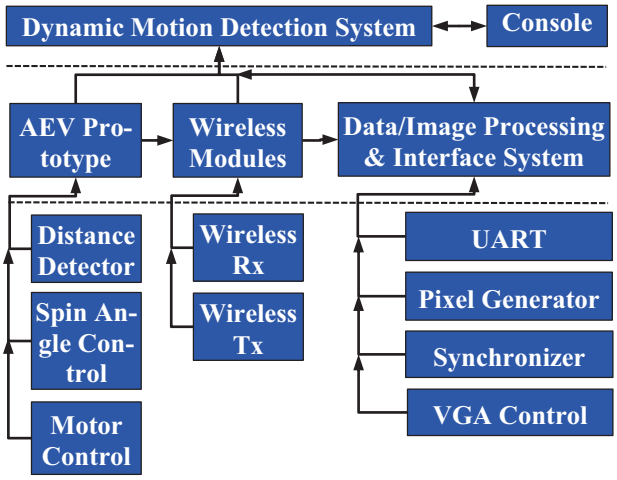



Figure 5. Architecture of the DMD System for iRTS

In order to cover a certain range of detection angle, the ultrasound sensor and image sensor modules are able to trace a single or multiple moving objects and obstacles. A servo module was introduced for spinning the sensor modules according to the driving speed of the AEV prototype. The servo module spins ±90º or ±45º depending on the AEV speed and the measured distance between the objects/obstacles and the AEV. The sensing data transmitted to and processed by a microcontroller, which is also interfaced with a wireless transmitter. The DMD system, then, is operated by the RPS algorithm programmed to the microcontroller.

A wireless module consisting of a pair of a transmitter and a receiver is built using two IEEE 802.15.4 modules. A data processing module is implemented with a microcontroller board. An associated interface system for the HIL simulation is designed with an FPGA. Any message generated for visual monitoring of the HIL simulation of the AEV prototype is transmitted to a VGA monitor connected to the console via a serial communication hardware implemented in the FPGA. The overview of the DMD design and key embedded hardware modules are illustrated in Figure 5. The generation of sampling data, wireless communication, and data/image processing are executed by the modules shown in Figure 5. The DMD system was designed as modular design architecture for offering the highest possible flexibility and compatibility with further extension. Three identified hierarchical layers convince for users to intuitively identify their embedded systems to apply to the proposed iRTS without spending time and resources to perform

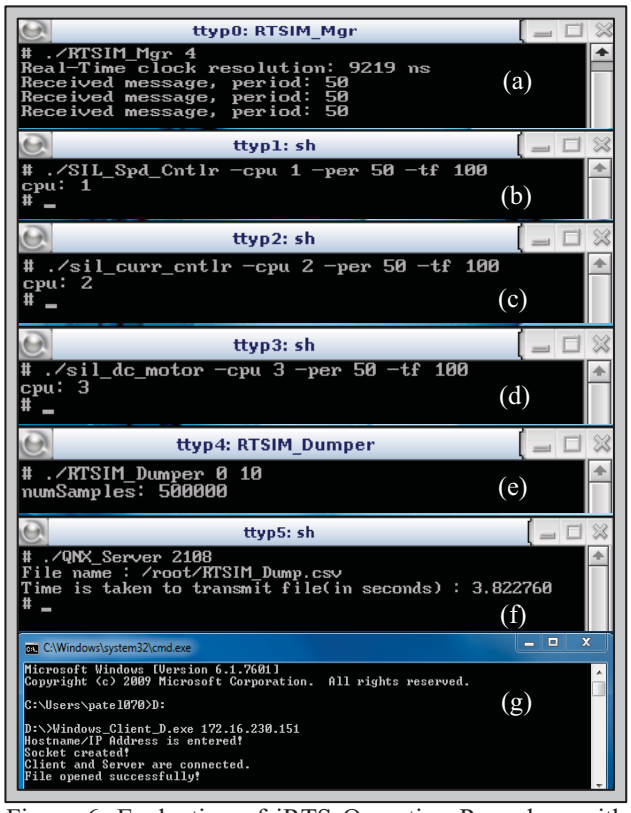



Figure 6. Evaluation of iRTS Operation Procedure with Console during the HIL Simulation: (a) Configure Models; (b, c, & d) Allocate Models to Cores (e) Configure Simulation Result, (f & g) Establish a Server-Client Connection for Transmitting the Results

accurate HIL simulation of their simulation models, hardware components, and even prototypes.

## V. Evaluation of the HIL Real-time Simulations with the iRTS

The iRTS evaluation has been performed for testing socket-based communication between the console and the iRTS via a wire (i.e., IEEE 802.3b) and a wireless (i.e., IEEE 802.11g) connections. Once the connection is established, models are transferred from the console to the iRTS. Once receiving the models, the iRTS launches a model management application, the iRTS manager, to deploy the necessary models. For instance, the HIL simulation performed with four models, including three SIL models—the modified speed controller, the current controller, and the DC motor, and the PWM virtual prototype implemented to FPGA. Since the PCI-based hardware interface is already assigned to a core, the PWM virtual prototype is not assigned to the core again. Thereby, three model acknowledgements are captured in Figure 6 (a). The iRTS manager configures all three SIL models' runs and then collects the simulation results in every 50 µsec for 100 simulation cycles as seen in Figure 6 (b), (c), and (d). In the meantime, the console must run a java application for dynamic plotting the HIL simulation results on the console screen. In order to execute the HIL simulation on the iRTS, another application, the iRTS dumper, must be launched. As seen in Figure 6 (e), the iRTS dumper initiates HIL simulation for 10 seconds and stores the simulation results to the output file, *.csv. Figure 6 (e) also shows that the iRTS dumper generated 500,000 samples during the 10 second HIL simulation. The HIL simulation results in the output file, *.csv, are transmitted to the console through the dedicate port on iRTS server (i.e., 2108). Figure 6 (f) illustrates the results were transmitted to the console for 3.822760 seconds. On the console side, IP address of the iRTS is used for creating a socket to communicate. This procedure completes a server-client connection between the iRTS and the console.

The results of determining the access time of the shared memory, PCI, and ADC in the PCI are shown in Figure 7 (a). The amount of time to access the shared memory is an average of 0.1486 µs. The time access the PCI is an average of 1.7889 µs, due to the bus used to access the PCI. The ADC in the PCI was identified as a critical path with an average of 13.2093 µs delay, which limits the faster model execution for the HIL simulation. Figure 7 (b) proves that the resolution of the real-time clock on the iRTS is set to 10 µs. Since the most reliable time to consistently operate at the same timing interval was identified as 50 µs, the model execution on the iRTS operates at a 50 µs for the real-time HIL evaluation.

In addition to measuring the hardware and shared memory access time, the execution time of the SIL models were measured to verify that the models are capable


Figure 7. Evaluation of iRTS Operations after the HIL Simulation: (a) Analysis of Operational Times; (b) Reference Clocking; and Execution Timing Analysis for (c) Speed Controller, (d) Current Controller, and (e) DC Motor subsystems

of executing within the 50 μs step time boundary. Figure 6 elicits the execution time measurement results. The times shown in the figures are the results of the difference between the real time clock at the beginning of the model step and after the model step was completed. The idle time for each model was also measured to account for the total execution time for each time step.

The results illustrated in Figure 7 demonstrate that the 50 μs step time is a viable step time due to no model requiring more than 7 μs to complete a time step. Figure 7 (a) illustrates that the speed controller requires an average of 792 ns to complete a time step. Figure 7 (b) clarifies that the current controller requires an average of 631 ns to complete a time step. Figure 7 (c) proves that the DC motor requires an average of 6158 ns to complete a time step, which is the largest time step, but well under the 50 μs time step utilized by the iRTS manager for scheduling the model execution time.

## VI. Conclusions

An iRTS platform for AEV is introduced for rapid and intuitive management and accurate real-time environment for research and education. The iRTS platform is also capable of executing both SIL and HIL simulations. In addition, the iRTS platform for developing AEV prototype is introduced. The proposed platform efficiently manages distinct hardware models and physical components on an AEV prototype throughout a hierarchical design procedure. In particular, the iRTS offers wireless capability for users to interconnect various embedded hardware components and software applications to the AEV prototype via additional wireless connections. A preliminary version of the MDM system is developed by integrating external embedded hardware modules interfaced to the iRTS platform without wire. In particular, the proposed iRTS efficiently schedules, distributes, interfaces, and simulates under developing software and hardware models on top of the AEV prototype. Furthermore, the iRTS provides sufficient wireless connections to multiple hardware prototypes via standardized wireless communication protocols. The HIL simulations of the AEV satisfy real-time constraints, such as 0.3% HIL simulation error, 7 μs HIL simulation time resolutions, and 2.5x acceleration speed of the motor control.

## References

[1] P. Menghal and A. Jaya laxmi, "Real Time Simulation: A Novel Approach in Engineering Education," in Proceedings of the IEEE Int. Conf. on Electronics Computer Technology, pp. 215-219, 2011.

[2] R. McNeal and M. Belkhayat, "Standard Tools for Hardware-in-the-Loop (HIL) Modeling and Simulation," in Proceedings of the IEEE Electric Ship Technologies Symposium, pp. 130-137, 2007.

[3] B. Suh and S. Berber, "Rendezvous points and routing path-selection strategies for wireless sensor networks with mobile sink," IET Journals & Magazines Electronics Letters, 52, (2), pp. 167-169, 2016.

[4] H. Salarian, K. Chin, and F. Naghdy, "An energy-efficient mobile sink path selection strategy for wireless sensor networks," IEEE Trans. on Vehicular Technology, 63, (5), pp. 2407–2419, 2014.

[5] C. Dufour, S. Abourida, and J. Belanger, "Hardware-in-the-Loop Simulation of Power Drives with RT-LAB," in Proceedings of the IEEE Int. Conf. on Power Electronics and Drive Systems, pp. 1646-1651, 2005.

[6] R. Dhole et al, "Smart traffic signal using ultrasonic sensor," in Proceedings of the Int. Conf. on Green Computing Communication and Electrical Engineering, pp. 1-4, 2014.

[7] O. Mohammed, N. Abed, and S. Ganu, "Real-Time Simulations of Electrical Machine Drives with Hardware-in-the-Loop," in Proceedings of the IEEE Power Engineering Society General Meeting, pp. 1-6, 2007.

[8] Zun Xu; Lili Liu, "The Application of Temperature Sensor in Automobile," in Proceedings of the Int. Conf. on Control, Automation and Systems Engineering, pp. 1-4, 2011.

[9] D. Georgoulas and E. In-Motes, "A Real Time Application for Automobiles in Wireless Sensor Networks," Measurement Science Review, Vol.3, No. 5, pp.158-166, 2011.

[10] L. Cheng and Z. Lipeng, "Hardware-in-the-Loop Simulation and Its Application in Electric Vehicle Development," in Proceedings of the IEEE Vehicle Power and Propulsion Conference, 2008.

[11] P. Menghal and A. Jaya Laxmi, "Real Time Simulation: A Novel Approach in Engineering Education," in Proceedings of the IEEE Int. Conf. on Electronics Computer Technology, pp. 215-219, 2011.

[12] C. Dufour and J. Belanger, "A PC-Based Real-Time Parallel Simulator of Electric Systems and Drives," in Proceedings of the Int. Conf. on Parallel Computing in Electrical Engineering, pp. 105-113, 2004.

[13] R. Sam, M. Tan, and M. Ismail, "Quad-Copter using ATmega328 Microcontroller," in Proceedings of the Int. Conf. on Electrical Machines and Systems, pp. 566-570, 2013.

[14] Baogui Zuo, "Research on computer simulation theory and experimental verification technology of the working process of automobile airbag," in Proceedings of the Int. Conf. on Management, Education, Information and Control, pp. 625-630, 2015.

[15] Yong Chen, "Application of 5 Kinds of Simulation Analysis in Automobile Engineering," Science and Technology Review, Vol. 25, No. 17, pp. 65-73, 2007.

[16] Chan, Chris, "High-Performance PCI Card: Main/ Lifestyle Edition," New Straits Times: 10. 2007.

[17] L. Junsong, et al, "FPGA Based Wireless Sensor Node with Customizable Event-Driven Architecture," Journal on Embedded Systems, 2013.

# SESSION

# EMBEDDED SYSTEMS, NOC, VLSI, REAL-TIME SYSTEMS, NOVEL APPLICATIONS AND TOOLS

## Chair(s)

**TBA**

ISBN: 1-60132-433-2, CSREA Press ©

ISBN: 1-60132-433-2, CSREA Press ©

# Overview of Assertion-Based Verification and its Applications

Zhihong Ren and Hussain Al-Asaad
Department of Electrical and Computer Engineering
University of California—Davis

*Abstract—Functional verification is a critical and time-consuming task in complex VLSI designs. There are two main challenges to functional verification: the first is to insure that the input stimulus can control the function spots inside the design and the second is to insure that the errors can be observed at the design output(s). Over the years, assertion-based verification techniques have been playing a more important role as part of functional verification methodologies. In this paper, we provide an overview of concepts, benefits, languages, and applications of assertion-based verification.*

Keywords—Design verification, assertion-based verification, assertion languages.

## I. Introduction

It is a fact that VLSI designs are getting increasingly more complex as times goes on. To maintain a competitive edge in marketplace, designers are packing even more logic gates onto a single chip. Typically, such complex design is formed by integrating reused blocks or IPs. Such reuse would lead to a greater challenge in verification. Those blocks function correctly in old designs, but may fail to work when integrated together. The new design may violate the assumptions of the reused blocks. To insure that new designs meet the specified requirements, any design violations must be found and corrected by verification.

To address the functional verification challenges, Assertion-Based Verification (ABV) has become a mainstream methodology [1]. ABV benefits from both theoretical research on formal methods and industrial practice in hardware verification as well as software engineering. ABV provides new capabilities in finding bugs, and of most importance, fits well with existing design and verification flows. With assertion languages standardized and well supported by verifications tools, assertions can bring immediate benefits to most verification projects.

The paper is organized as follows. In the sections II, we clarify some basic concepts used in design verification. In sections III, we explain benefits and effects from assertion-based verification. In section IV, we introduce three assertion-based techniques and compare their coverage and quality of verification. In section V, we mainly focus on how to implement ABV in the hardware design process. In the final section, we conclude the paper.

## II. Basic Concepts

### A. Linear Temporal Logic (LTL)

In theory, properties are often written as formulas of linear-time temporal logic (LTL). In LTL, one can encode formulas about the future of paths, e.g., a condition will eventually be true, a condition will be true until another fact becomes true, etc.

In typical hardware design, there is usually a special signal called the clock. The entire system synchronizes its actions along with the clock. Since synchronous design can be modeled as Kripke structures (a formal semantics representing logic systems), researchers often wrote specifications in LTL in the early 1980s to express properties such as "hit this state when you receive that signal". Their basic idea is to consider signals as atomic propositions and simulation waveform as structures, and thus relations of signals over clocks can be expressed by LTL formulas.

### B. Formal Verification

Formal verification is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics. One approach and formation is model checking. This method automatically translates a hardware design into symbolic representations of a set of structures and then proves interesting properties of the design. Another approach is deductive verification. It consists of generating from the system and its specifications (and possibly other annotations) a collection of mathematical proof obligations, the truth of which imply conformance of the system to its specification, and discharging these obligations using either interactive theorem provers, automatic theorem provers, or satisfiability modulo theories (SMT) solvers. The growth in complexity of designs increases the importance of formal verification techniques in the hardware industry.

### C. Assertion-based Verification (ABV)

Assertion-Based Verification is a methodology for improving the effectiveness of a verification environment. Designers use assertions to capture specific design intent and, either through simulation, formal verification, or emulation of these assertions, verify that the design correctly implements that intent.

## III. Advantages and Influences of ABV

### A. Weakness of the traditional verification methods

Coverage-driven verification is a verification methodology in which coverage planning precedes the rest of the verification process [2]. In this method, test vectors are used to find the errors in the design. The method can notify the designers that there exist bugs in the design (without information about their location). Designers have to look into the entire system and trace waveforms cycle by cycle in order to find where the bug is located.

### B. Advantages of ABV

1. Verification efficiency: Assertions can be verified by different checking tools both in formal verification or

simulation environments. This improves the verification quality of many corner cases without stopping or fixing the existing problems.

2. Observability and controllability: In the traditional way any design is verified via the use of a simulator with the help of test bench and input stimuli. In order to identify a bug in the design, it is required to generate a proper input, which will also propagate the wrong value to the output port of the system from the position of the actual problem. This is commonly known as the Black-Box Verification. The assertion-based verification catches the wrong output at the point of its occurrence and reports the error. In this way it not only improves the observability of the system but also improves the overall controllability over the system via reducing the search area for finding out the cause of the bug. This requires the use of White-Box Verification.
3. Modules compatibility: The key concern in integration test is the trade-off between testing costs and testing quality. If verification engineers integrate a large amount of modules at one time, it is hard to do a complete test, and more importantly, it is hard to locate bugs. In nowadays IC design process, a complex design is usually composed of different modules developed by different designers. So, it is necessary to monitor the interfaces between modules. Assertions can verify the compatibility of all the modules by checking the properties of interfaces.
4. Reusability: Like modules reusability, assertions and coverages themselves are reusable. Different designs may share the same modules and interfaces. So the properties on interfaces can be checked with the same assertion checkers.

### C. *Influences of ABV*

1. Functional verification plays more and more important role in complex VLSI design, especially for the detection of potential bugs in interfaces among different modules. Assertions allow the verification engineers to define design properties in higher description level, and this is more concise and natural than hardware description languages. And since assertions are used in the design process, we could start the formal verification in an early state.
2. Properties that are defined by assertions can be reused. This is the case since there is no necessary relationship between assertion description and implementation. So the user can easily reuse those properties descriptions in various designs.
3. The coverage analysis in traditional verification methodology is based on line coverage. The ABV methodology includes functional coverage properties (A functional coverage property is true if certain functionality has been exercised by a test).

## IV. Standard Assertion Languages and Checkers

The most common standard assertion languages are the SystemVerilog Assertions (SVA) and the Property Specification Language (PSL). These languages are available to capture the functional specification of the design including assumptions, obligations, and invariants. The Open Verification Library (OVL) is not a language but a library of predefined checkers written in VHDL or Verilog.

### A. *SystemVerilog Assertions (SVA)*

SystemVerilog adds features to traditional Verilog to specify assertions of a system. Also, assertions can be used to provide functional coverage and generate input stimulus for validation. There are two kinds of assertions: immediate and concurrent. To ensure the integrity of the program, SystemVerilog also provides the bind function in an assertion, so we can bind the assertion module with design modules.

*a) Immediate assertions:* An immediate assertion is a test of an expression performed when the statement is executed in the procedural code. The expression is non-temporal and treated as a condition similar to an "if statement". Its basic syntax structure is shown in the Fig.1. The expression here is a boolean variable. When it is equal to X, Z or 0, the assertion module will report errors.

```
procedural_assertion_statement::=
...
| immediate_assert_statement
immediate_assert_statement::=
assert (expression)  action_block
action_block::=
statement_ or NULL
| [statement]else statement
```

Fig.1. Immediate Assertions [5].

*b) Concurrent Assertions:* A concurent assertion describes behavior that spans over time. The basic syntax is shown in Fig. 2. In a concurrent assertion, evaluation is performed only at the occurrence of a clock tick. An expression used in an assertion is always tied to a clock definition. The sampled values are used to evaluate value change expressions or boolean subexpressions that are required to determine a match of a sequence. The keyword property distinguishes a concurrent assertion from an immediate assertion.

*c) Bind:* In order to separate design and verification conveniently, SVA provides Bind function. That means one can write all the assertion he or she wants to write in a separate file and using bind, he or she can bind the ports of his assertion file with the port/signals of the RTL in his testbench code. A bind feature can be used in module, interface and compilation unit scope. Thus we can achieve the following:

- Even if the design code has been slightly changed, verification engineers can verify the design based on the original environment.
- The verification IP can be easily and quickly added to a submodule.
- There is no need for a syntax change for the assertions inside property function, which ensures the stability and reliability of the program.

*d) SVA checker library:* the library consists of two parts:

- SVA consists of checkers that have the same behavior and controls as those in the Open Verification Library (OVL). These checkers have additional features such as coverage.

- SVA advanced checkers contains checkers that generally verify more complex behaviors. These have similar controls to the OVL checkers, but, in addition, they allow selecting the sampling clock edge(s), posedge or negedge. Coverage is also provided.

```
assertion_statement::=
concurrent_assertion_statement
concurrent_assertion_item::=
[ block_identifier: ] concurrent_assertion_statement
concurrent_assertion_statement::=
assert_property_statement
assert_property_statement::=
assert property (property_spec) action block
concurrent_assertion_item_declaretion::=
property_declaration
...
property_declaration:=
property property_identifier[ ( [ list_of_formals ] ) ];
{assertion_variable_declaration}
property_spec;
endproperty [:property_identifier]
list_of_formals::= formal_list_item {, formal_list_item}
property_spec::=
[clock_event] [disable iff (expression_or_dist)] property_expr
property_expr::=
sequence_expr
[ (property_expr)
| not property_expr
| sequence_expr |-> proper_expr
| sequence_expr |=> proper_expr
| if (experssion_or_dist) property_expr [else property_expr ]
assertion_variable_declaration::=
data_type list_of_variable_identifier;
property_instance::=
ps_property_identifier [ ( [actual_arg_list] ) ]
```

Fig. 2. Concurrent Assertions.

### B. Property Specification Language (PSL)

PSL is a separate language specifically designed to work with many HDLs and their expression layers [12]. As a result, PSL cannot be written directly as a part of any HDL. However, PSL properties can be attached to HDL models using binding directives, and tools can support PSL inclusion in HDLs. One of the domains addressed by PSL is that of system verification. A good number of system design houses (e.g. IBM, Intel) employ this pre-RTL methodology, where a high-level description of the system is modeled in an FSM form and verified against the architectural requirements [6]. PSL provides special support for this powerful verification methodology using the GDL (Generic Definition Language) flavor. A different application of PSL will be its extension to analog and mixed signal domains. A working group sponsored by the EU is presently pursuing the definition of such extensions to PSL [7]. Yet another pressing application is the verification of asynchronous designs, and work on extending PSL to support such design style is underway. It is conceivable that more domains, applications and language flavors for PSL (which by design is flavor-extendible) will come up in the near future. One such creative application of PSL is its use for aerospace control applications; an earlier one (where the base Sugar [13] language was actually used) is for validation of railway interlock protocols.

PSL provides the capability to write assertions that range from system-level in various kinds of systems, and down to RT level. PSL has a structure of multiple abstraction layers and a rich set of operators that can be used at different levels of abstraction. The low-level layer of PSL, which governs the application domain, can be easily adjusted to many applications and design languages. Moreover, the application layer can be even extended or replaced by a different layer to support new applications. In summary, PSL is a multi-purpose, multi-level, multi-flavor assertion language. In contrast, SVA is tightly connected to the SystemVerilog language.

### C. Open Verification Library (OVL)

The OVL is composed of a set of assertion checkers that verify specific properties of a design [14]. These assertion checkers are instantiated in the design establishing a unifying methodology for dynamic and formal verification. OVL assertion checkers are instances of modules whose purpose in the design is to guarantee that some conditions hold true. The syntax of every Assertion checker consists of one or more properties, a message, a severity and coverage.

- *Properties* are design attributes that are being verified by an assertion. A property can be classified as a combinational or temporal property.
- A line of *Message* is a string that is displayed in the case of an assertion failure.
- *Severity* indicates whether the error captured by the assertion library is a major or minor problem.
- *Coverage* indicates whether or not specific corner-case events occur and counts the occurrences of specific events.

Assertion checkers address design verification concerns and can be used as follows to increase design confidence:

- Combine assertion checkers to increase the coverage of the design (for example, in corner-case behavior or interface protocols).
- Include assertion checkers when a module has an external interface, in which case, the assumptions on the correct input and output behavior should be guarded and verified.
- Include assertion checkers when interfacing with IP modules, since the designer may not be familiar with the module description as it may be an IP core used for re-use, or may not completely understand the module, in which case, guarding the module with OVL assertion checkers may prevent incorrect use of the module.

OVL assertion checkers are partitioned into the following checker classes:

- Combinational assertions: the behavior is checked with combinational logic.
- 1-cycle assertions: the behavior is checked in the current cycle.
- 2-cycle assertions: the behavior is checked for transitions from the current cycle to the next.
- n-cycle assertions: the behavior is checked for transitions over a fixed number of cycles.

### D. Comparison

All the languages have similar ability to express assertions. However, they still differ in terms of finer details and in their support mechanisms in various tools. A comparative look at the support for assertions between OVL, PSL and SystemVerilog is shown in Table 1.

As described earlier, OVL is designed as a library comprising a set of predefined modules, which can be instantiated within the design to get the desired assertion

effect. As it is not an enhancement of any language, this library based mechanism has both syntactic and semantic limitations. The assertion statements are basically module instantiation where the actual definition of the assertion is written in the module. So the future enhancement of the assertion feature will be limited to only the addition of new port and parameter and also it is not very easy to understand the meaning and usage of these ports and parameters without the help of the language reference manual.

Table 1. Comparison of assertion techniques [5].

| Parameters | SVA | PSL | OVL |
|---|---|---|---|
| Property declaration | Y | Y | Y |
| Property check | Y | Y | Y |
| Constraint specification | Y | Y | N |
| Sequential mechanism | Y | Y | Y |
| Functional coverage | Y | Y | N |
| Clock synchronization | Y | Y | Y |
| Immediate assertion | Y | N | N |
| Declaration assertion | Y | Y | N |
| Program assertion | Y | N | Y |
| Built-in function | Y | Y | N |
| Implementation | Easy | Easy | Not easy |

PSL, on the other hand, is a totally different language. It was not easy to directly encapsulate the whole language within Verilog and though it is permitted to use the PSL constructs directly in the RTL code, it is always used by lot of people using pragma mechanism (as an embedded comment that starts with "PSL") to keep the RTL design tool independent of the assertion support.

SystemVerilog is defined as an extension of the Verilog language including the features of assertion and functional coverage. This would help both the designers and the verification engineers to write tool independent design and verification modules in the same language without adding any tool specific pragma or semantics. And moreover this has made both the modules understandable to everyone. It is a pure HDL where both the Hardware Description and its associated Verification specification (read "design intent") are covered together.

## V. APPLICATIONS

### *A. Verification hot spots*

Verification hotspots [16] are areas of concern within the design that are difficult to verify using traditional simulation-based methodologies. A verification hot spot is typically difficult to verify because it contains a great number of sequential states; it is deeply buried in the design, making it difficult to control from the inputs; it has many interactions with other state machines and external agents; or it has a lot of the above characteristics. Therefore, a verification hot spot cannot be completely verified by simulation alone—the amount of simulation and the difficulty of creating a sufficient input vector set are simply prohibitive.

The value of a verification hot spot solution is that it captures and makes explicit the knowledge of how to effectively use formal verification and simulation together to fully verify a section of logic [3]. In this way, the expert knowledge of one user can be transmitted to others. This knowledge includes such items as how to capture the right assertions for a verification hot spot; what mix of formal and simulation analysis is appropriate; what input constraining methodology is effective for the verification hot spot; what metrics need to be measured and what the metrics goals need to achieved.

**a) Resource logic control**: Computation resources, system on-chip buses, inter-connections, buffers and memories are logic structures usually controlled by arbiters and complex control logic. When creating directed tests for the design, verification researchers tend to focus on the high-level specifications. They seldom stress the corner cases of these resource control logic, as a result, some problematic scenarios are not observed.

Properties:

- Ensure the requests are generated correctly based on the mask, the weight, the credit scheme and so on. This is done by using procedural descriptions to generate the "glue logic" associated with the corresponding assertions.
- Ensure the arbitration scheme is correct. For straightforward arbitration schemes (such as round-robin, priority, least-recently-used and so on) the arbiter checker from an assertion library can be used directly.
- Ensure the resource (bus, inter connect, memory) is addressed by only one master at a time. This can be done with an assertion language or with the mutex checker from an assertion library.
- Ensure the resource is de-allocated before it is allocated again. Semaphores lock and release the common resources that do not allow multiple simultaneous accesses. Such locks should be set before an access.

Methodology:

If the design is well modularized (i.e. the targeted control logic is self-contained), formal model checking can be applied early even before the simulation environment is ready.

In a typical system, resources can be requested at any time and allocated in any sequence. Verifying all possibilities with simulation is tedious (if not impossible). Alternatively, by capturing the environmental assumptions as assertions, formal model checking can analyze exhaustively all possible scenarios and detect any potential resource conflict. Once the system simulation environment is ready, these assumptions can be validated in the system environment.

**b) Design Interface**: Inter-module communication and interface protocol compliance are infamous for causing a lot of verification failures. When design teams first integrate all the blocks together for chip-level simulation, the blocks usually fail to "talk" with each other. To catch this problem early, assertion-based protocol monitors are added to instrument the on-chip buses and the standard and common interfaces.

Properties:

- Ensure the correctness of the protocol on all of the bus interfaces, especially when there are different modes of operation. As the popularity of standard assertion languages increases, more assertion monitors for standard bus interfaces become available.
- Ensure the transparency of the interfaces. This property ensures that the interface does not lose transactions. For example, in a bus bridge design, transactions on the master interface should be reflected at the target interface. These requirements can be expressed with an assertion language using temporal properties.

Methodology:

Monitors written with an assertion language can be used with both simulation and formal verification. In addition, assertion monitors track simulation coverage and statistics information, which provides a means for gauging how thoroughly bus transactions have been stimulated and verified.

Assertion monitors are useful for verifying internal interfaces as well. Such interfaces have ad hoc architectures, so they are particularly prone to mistakes caused by misunderstanding and miscommunication.

To validate the transparency of an interface, we must ensure that the transactions are interpreted and transferred correctly. Simulation with assertions can validate the one-to-one mapping of the basic transactions. However, to be thorough, we have to ensure the illegal, error and retry transactions are also being handled correctly.

**c) Finite state machine**: For verification purposes, we classify finite state machines (FSMs) into two categories: interface FSMs and computational FSMs. Interface FSMs use I/O signals with well-defined timing requirements. Examples of interface FSMs are bus controllers, handshaking FSMs and so on. Computational FSMs do not involve signals with explicitly defined timing requirements. Examples of computational FSMs are FSMs for flow charts, algorithmic computations, and so on.

Properties:

- Typically, specifications for interface FSMs are derived from waveform diagrams. So, it is crucial to ensure that the FSM samples the input signals within the time period correctly and the response signals are asserted within the output timing specification. It is natural to express these timing requirements with an assertion language using temporal properties.
- Typically, specifications for computational FSMs are derived from control flow graphs, which are common in engineering documents and standard specifications. To improve performance and/or simplify implementation, a flow graph might be partitioned, flattened, re-timed or pipelined into multiple-FSMs. But, regardless of the performed optimization, they should still mimic the flow graph correctly. Implementing a control flow graph with assertions captures its specification in an "executable" from. These assertions then ensure all of the FSM's decisions and transitions are made correctly. Such assertions can be implemented with a mix of procedural description and temporal properties.

Methodology:

During simulation, assertions monitor activities inside the design and provide information showing how thoroughly the test environment covers the design's functionality. By capturing the properties of FSM using assertions, we can identify them easily, exercise them completely, and collect cross-product coverage information during the simulation process. In addition, the implementation of the FSM also has a direct affect on the effectiveness of verification.

When several FSMs are interacting with each other, it is important to ensure that they do not get "trapped" in a corner-case behavior. Formal verification can be applied to check this situation. However, formal model checking is not efficient with complex properties, which are unbounded. Hence, the high-level timing requirements of an FSM may have to be decomposed and captured within several assertions.

**d) Data Integrity**: Devices such as bus bridges, DMA controllers, routers, and schedulers transfer data packets from one interface to another. Unfortunately, in a system-level simulation environment, data integrity mistakes are not readily observable. Usually, problems are not evident until the corrupted data issued. With ABV, assertions check the integrity of data along the entire data transfer path. A lost or corrupt data packet is detected immediately.

Properties:

- Ensure key registers (program counters, address pointers, mode/configuration registers, status registers) are programmed correctly and their data are never lost. Key registers contain "precious" data that must be consumed before being over-written. Assertions can ensure these registers are addressed, programmed and sampled correctly. An assertion language can capture the desired properties by probing into the registers hierarchically.
- Ensure data going through a queue is not lost or corrupted. Data transferred through a queue must follow the first-in first-out scheme without any alteration. The best way to capture this property is to use an approach that mixes both an assertion language and an assertion library. The assertion language readily captures the enqueue and dequeue protocol conditions. Instead of manually creating the data integrity assertion, we use a data integrity checker from the CheckerWare assertion library. The checker ensures that:
  1. No more than *n* transactions are ever stored in the queue.
  2. No outgoing transaction is generated without a corresponding incoming transaction.
  3. Data written to the queue is not corrupted.
- Ensure tagged data in the system are not lost or corrupted. In many designs (for example, data switches and routers), data are stored temporarily before being transferred. A *tag* is a handle on the data content. Structures that store and retrieve data can be verified using checkers (for example, the memory and score board checkers from an assertion library). The type of storage structure is not important. The objective is to ensure that the data stored in memory is not corrupted; the read/write ordering is correct and the address has no conflicts.

Methodology:

End-to-end simulation-based verification methodologies transfer many data packages so that many of the data integrity assertions are thoroughly exercised. However, simulation typically fails to stress test storage elements by filing up the FIFOs/memories. So, pseudo-random simulation environments should be guided by coverage information from the assertions. This can be done manually (by fine tuning the random parameters) or automatically (using an active test bench that monitors the assertions).

To compound the issue, multiple queues and data streams might transfer and retrieve data at unpredictable intervals. Simulation alone cannot verify all of the corner-case scenarios; there are simply too many of them. So, formal model checking is used to target the remaining data integrity assertions. It can explore al legal combinations of transfer events to ensure the data are never corrupted. When we

examine the bugs found with this methodology, they are all related to complex combinations of events initialized by different data transfers.

*B. Simulation based SVA verification of UART*

The work presented in [15] summarizes seven steps of formal verification planning process, which are demonstrated by Harry Foster [17]. Because of the disadvantages of formal verification, simulation is still the main method to do functional verification currently. To apply this method in simulation-based verification, we simply reduce and adjust this method to five steps as follows [4]:

- List design interfaces.
- List the inside functional spots.
- List verification requirement for interface.
- Formalize the properties with SVA.
- Define coverage spots.

Following the five steps above, we can easily find the design errors from the waveform window or just from the log file of the simulation. When it comes to using the SVA to describe the desired properties, it is quite easy to debug those errors. This is because every individual assertion has a unique name, and we usually embed those assertions into different locations of the source code. By examining the log file, we can find both which assertion fails and the exact location of this assertion in the source code. The assertion is close to where the error occurs, thus we can easily find and debug this error. Further, assertions usually are just used in simulation and do not make up hardware resource, therefore we just add as many assertions as possible into the design.

To investigate this method in real designs, we approach it in the functional verification for an RTL model of a universal asynchronous receiver/transmitter (UART)—a hardware module that translates data between parallel and serial forms.

*a) List the inside functional spots*

Fig. 3 depicts the interface signals of the UART transmitter, which consists of UART controller, data register, shift register and count register. In this step of verification, we consider this design as a "black box", which means that we do not care about its implementation, while just check whether the interface signals accord with the requirement of the design specification, especially for the timing requirement. In this case, we check whether the test bench generates valid stimulus for the design under verification.

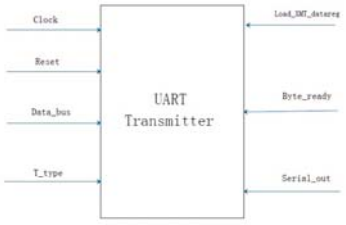



Fig. 3. UART interface model.

*b) List the inside functional spots*

In this step, we list the verification requirements for the candidate module. These are the properties and the features inside the module, which need to be verified. One crucial component for nearly every design is the control block, which is also where some vital design errors occur. Thus control block is what we should devote great effort to verify. In the UART transmitter, the control block is implemented by FSM, shown in Fig. 4. Therefore to examine if the FSM works exactly the same as what the design specification describes is the key step for its functional verification. The general verification for a FSM is to assure the transitions between states are correct and follow the timing requirement.

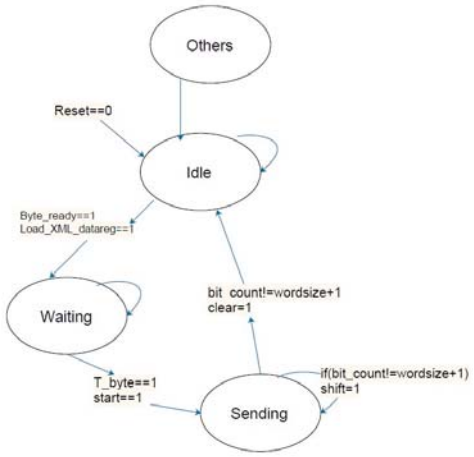



Fig. 4. State transition graph of the UART transmitter.

Table 2 shows the state transitions that need to be verified. The first row is the current state, while the first column indicates the next state. Each check mark indicates a state transition, which should be verified. Besides those simple pairs of state transitions, extra checks should also be conducted, such as if a complete state transition flow is performed.

Table 2. State transactions that needed to be verified.

| State transactions | | Current state | | | |
|---|---|---|---|---|---|
| | | Other | Idle | Waiting | Sending |
| Next state | Other | | | | √ |
| | Idle | √ | | | |
| | Waiting | | √ | | |
| | Sending | | | √ | |

*c) List verification requirement for interface*

In this step, we list the verification requirements for the interface signals of each module. As to the UART transmitter, we check whether it functions well with other blocks sharing common signals with it. Transmitter is the top module of the design, thus we examine if the test bench generates valid input stimulus. To ensure the UART transmitter works correctly, the test bench should provide control signals in a proper sequence, as shown in Table 3.

Table 3. Interface signal features to be verified.

| Interface Features | Signals and their features | |
|---|---|---|
| | Revenant Signals | Expected Features |
| 1 | Byte_ready, Reset | Byte_ready =1 no less than one cycle after reset is released |
| 2 | T_byte, Byte_ready | T_byte =1 no less than one cycle later when Byte_ready is asserted |
| 3 | Load_XML_datareg, data_bus | When load_XML_datareg is asserted, data_bus must hold valid data |
| 4 | … | ... |

This step is significant especially for a multi-module complex design. After the unit level verification, we still need a system level verification, and that is why we separate this step from previous step. Differentiating interface signals'

properties from each other allows us to focus on checking the coordination/integration of different modules at system level.

*d) Formalize the properties with SVA*

In this step, we convert the natural language requirements into a set of formal properties using assertion language or assertion checkers. We can instantiate assertions from library or express them with assertion language (SVA).

In the latter case, we describe the functions to be verified with the combination of several logic events, which can be described using SVA key word "sequence". The simplest logic event is a change on the value of a signal; a logic event can also be a change on the value of a Boolean expression. Key word "property" in SVA is to represent complex behaviors of a specific function. Property is the unit being verified during simulation, and the key word "assert" is used to check the property.

Fig. 5 shows an example of SVA checker, which is given valid control signals sequences. First change the checker to "sending" state, then keep it in this state for several cycles and stay in state "idle" at the end.

*e) Define coverage spots*

In this step, we define properties as functional coverage spots. By using this method, we can analyze the assertion coverage as an indicator of the functional coverage. Typically, there are two functions of an assertion: one is serving as checker for a specific function; the other is being used for statistical analysis for coverage. To demonstrate this, take the UART for example. We define all the properties mentioned above as coverage spots. An example of how to describe an assertion and how to define it as a coverage spot is illustrated in Fig. 5.

We can check whether the transitions between pairs of states are correct, and can also get the information concerning how many of these state transitions are conducted. In fact, this is also a kind of FSM coverage analysis. By analyzing the coverage report, we can judge whether the verification is sufficient. After defining assertion coverage spots, we can get the statistical analysis of every assertion, such as the number of over all attempts, matches, vacuous matches, and fails.

```
sequence s_finish_sending;
##1    ((state==sending)   [*9])   ##1      (state==idle);
 endsequence
//Define a sequence

property p_finish_sending;
@(posedg Clock)
(reset_)&&(state==sending)&&($past(state==waiting) |-> s_finish_sending;
endproperty
//Define a property

a_finish_sending:
assert property(p_finish_sending);
c_finish_sending: cover
property(p_finish_sending);
 //Define a coverage spot
```

Fig. 5. An example of a SVA checker.

Further, different severity levels can be attached to assertions. If some assertions fail, according to their severity levels, different measures are adopted, such as terminating the simulation or just displaying the fail messages and keeping on the simulation.

## VI. Conclusion

In this paper, we discussed assertion-based verification and its simple application in integrated circuits. Because of the advantages of ABV, verification teams can easily check the corner cases and locate the design errors. Furthermore, assertion coverage reports can be directly used to analyze the functional coverage metrics of a design.

We also introduced three kinds of assertion languages and checkers, including SVA, PSL and OVL. Those languages and checks are widely used in complex VLSI design verification processes. At a high level, the choice between these languages and checkers may depend on interoperability and marketing decisions. Finally, we presented several applications of ABV.

## References


[1] B. Turumella and M. Sharma, "Assertion-based verification of a 32 thread spareTM CMT microprocessor", *in DAC 08: Proceedings of the 45th annual Design Automation Conference.* New York, NY, USA: ACM, 2008, pp. 256-261.

[2] Yuan, Jun, Carl Pixley, and Adnan Aziz, "Constraint-based verification" Springer Science & Business Media, 2006.

[3] Yeung, Ping, and Kenneth Larsen, "Practical assertion-based formal verification for SoC designs." *System-on-Chip, 2005. Proceedings. 2005 International Symposium on*. IEEE, 2005.

[4] Li, Yangyang, et al, "A study on the assertion-based verification of digital IC," *Information and Computing Science, 2009. ICIC'09. Second International Conference on.* Vol. 2. IEEE, 2009.

[5] Sonny, Ashley T, and S. Lakshmiprabha, "OVL, PSL, SVA: Assertion based verification using checkers and standard assertion languages." *Advanced Computing and Communication Systems (ICACCS), 2013 International Conference on.* IEEE, 2013.

[6] Eisner C, Shitsevalov I, Hoover R, et al. "A methodology for formal design of hardware control with application to cache coherence protocols". *Proceedings of the 37th Annual Design Automation Conference.* ACM, 2000: 724-729.

[7] Y. Wolfsthal, "EU-Sponsored Deployment of PSL", *PSL/Sugar Consortium Meeting*, DATE 04, Feb 2004

[8] Foster, Harry, Kenneth Larsen, and Mike Turpin, "Introduction to the new accellera open verification library." *Proceedings of the Conference on Design and Automation and Test in Europe*: DATE. Vol. 6. 2006.

[9] Foster, Harry, Kenneth Larsen, and Mike Turpin, "Introduction to the new accellera open verification library." *Proceedings of the Conference on Design and Automation and Test in Europe*: DATE. Vol. 6. 2006

[10] Foster, Harry. "Assertion-based verification: Industry myths to realities (invited tutorial)." *Computer Aided Verification. Springer Berlin Heidelberg,* 2008.

[11] Cohen, Ben, Srinivasan Venkataramanan, and Ajeetha Kumari. "Using PSL/Sugar for formal and dynamic verification: Guide to Property Specification Language for Assertion-based Verification." *VhdlCohen Publishing*, 2004.

[12] SystemVerilog 3.1a Accellera's Extensions to Verilog, Accellera, Napa, California, 2002

[13] Property Specification Language Reference Manual, Accellera, Version 1.1, 2004

[14] Research.ibm.com. "PSL/Sugar". N.p., 2016. Web. 29 Feb. 2016.

[15] Ping Yeung and Sundaram Subramanian, "Applying Assertion-Based Formal Verification to Verification Hot Spots", *Mentor Graphics Technical Library,* October 25, 2007

[16] Ping Yeung, "Functional Verification of ARM-based Platform Design with Assertion-Based Verification Methodology", *ARM Developers Conference*, 2004.

[17] Harry Foster, "Integrating Formal Verification into a Traditional Flow", *Mentor Graphics White Paper*, 2006.

# Development of Data Logger for Non-Wired Sensing and Recording of Small Lateral paths

**Tongjun Ruan , Robert Balch**
Reservoir Evaluation and Advanced Computational Technologies Group
Of Petroleum Recovery Research Center , New Mexico Tech., Socorro, NM, USA

**Abstract:** *A 3-Axis Gyroscope and 3-Axis Accelerometer data logger was designed for non-wired sensing and recording of the small diameter laterals where the wire or wireless communication with ground/surface is impractical or not feasible. This is accomplished by measuring 3-axis angular rate and 3-Axis acceleration, storing the data on the data logger, offloading data to an onsite laptop computer and then back-calculating the path with Strapdown Inertial Navigation Algorithm.*

**Keywords:** Data logger; 3-Axis gyroscope; 3-Axis accelerometer; Strapdown Inertial Navigation; Radial Drilling;

## 1 Introduction

The path of small diameter laterals has been a mystery due to the very limited space and harsh environments. Radial Drilling is a versatile technology with a range of applications from well enhancement, increased injection rates in disposal wells, deep acid penetration, directed fracturing applications and other applications.

The drilling is accomplished through a high pressure water jet and can extend up to 300 ft horizontally from a vertical well-bore. Since the water jet is deployed on the end of a 300 ft rubber hose attached to a coiled tubing unit there is no method to directly measure the path of the radially jetted lateral since no wire or other direct connection is possible. The small size of the hose and drilling tip (28.6mm Diameter) and the shoe which turns the hose horizontal in the well-bore, having a short turning radius, limits the size of any measuring device to about 15 mm length by 20 mm diameter.

In addition the measurements must be stored directly within the device, and survive a hostile environment with high temperatures, solvents, and high g impacts.

It is very difficult to directly measure the path (direction and distance) underground. Usually accelerations in 3 directions are measured and then the velocity is calculated by integrating acceleration, distance is calculated by integrating the velocity based on Newton's laws. The datalogger may rotate during recording data, the recorded accelerations will be related "body frame" which makes calculating the path in a world coordinate impossible, The data logger is designed to measure the angular rates and the accelerations in X, Y and Z directions and store the data into EEPROM during its going to and back through the radial well. The recorded data will be offloaded into an onsite laptop computer/PC, be processed and be used to calculate the path.

## 2 Hardware design

Due to the very limited space and harsh environments, the data logger must be small enough in size and can be put into the protecting steel shell with length less than 28mm and diameter less than 28.6 mm diameter.

### 2.1 Limited space

Considering the wall thickness of the protecting steel shell and two end caps' thickness, the device PCB must be less than 15mm length and 20mm diameter. Figure 1 shows the protecting tube/shell of data logger and the inside usable space.


*Figure 1. Steel Shell and its dimensions and ledge*

Steel shell size: 28.6 mm outer diameter X 28mm length , The house room is for data logger to stay during test, its diameter is 20 mm and its length is 15mm, due to the ledge where the diameter is 20mm. Figure 2 shows the house room with 25 diameter and can hold a PCB with dimension 20mm L. X 15 mm W..

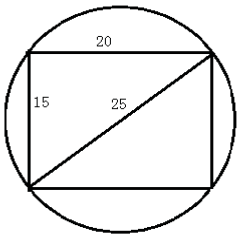



*Figure 2. The usable room for DataLogger*

## 2.2 Chips and Battery

Based on the requirements, we need :

1. A sensor with 3-Axis Gyro and 3-axis accelerometer to measure the angular rates and accelerations in X, Y, and Z directions;
2. A flash memory chip to store the data ;
3. MCU to configure the sensor and the flash memory, read data from the sensor and write data into the flash memory;
4. Battery to provide operating power;
5. Other chips needed to make the circuit work properly.

There are quite a few of sensors with 3-Axis gyro and 3-axis accelerometer from different companies. One of them is InvenSense's MPU9250, which is the company's second generation 9-axis(3-Axis gyro, 3-axis accelerometer and 3-axis compass) MotionTracking device of its world's first 9-AxisMotionTracking device designed for the low power, low cost, and high performance requirements of consumer electronics equipment including smartphones, tablets, and wearable sensors. This was chosen as the sensors in the data logger mainly due to following[1] :

- MEMS Digital output 3- axis angular rate sensor and 3-axis accelerometer
- Ultra low-power (9.3uA)
- Three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs, three 16-bit ADCs for digitizing the accelerometer outputs
- InvenSense provides Embedded MotionDriver (9/6-axis solution)[3], which is available in full source with a DMP binary image that is responsible for fusing and calibrating the data from the gyroscope and accelerometer.
- Small size, 3x3x1mm QFN package

Because the DataLogger will be sealed inside steel shell, the compass may not work properly and also because the MCU's limited program memory 8K Words and 1KB RAM not allow loading big DMP binary image, 6-axis solution was chosen, the 2334 bytes of DMP binary image must be programed into MPU-9250 chip before enabling the MPU-9250 to work). IS25LP128 serial flash memory was chosen to store the data in data logger due to[2]:

- 128M-bit/16M-bytes ( the largest in market, can store 2-hour data, discussed in section III )
- Industry Standard Serial Interface
- 256 bytes per Programmable Page
- Supports standard SPI
- 8-pin SOIC 208mil
- Lower operating voltage(Single 2.3V to 3.6V Voltage Supply)

Microchip's PIC16LF18250 was chosen to play the MCU role in the data logger, mainly because following[7] :

- Its extra low power consumption,
- 8KW program flash memory
- Integrated MSSP with SPI and I2C.
- 14-pin SOIC
- Small size

All 3 chosen chips are small in size and meet the space limitation.

Lithium Coin battery LR1632 is a good choice for the data logger with 3V voltage and smallest size among Lithium coin batter(diameter =16mm and length =3.2mm) and 140mAH capacity.

| Name | PIC16LF1825 | MPU-9250 | IS25LP128 | Battery LR1632 |
|---|---|---|---|---|
| Size | 3.9x3.9mm | 3x3mm | 5x5mm | D 16 x L 3.2 mm |
| Voltage | 1.8-.6v | 2.4-3.6v | 2.3x3.6v | Li. 3v |
| Current | 75uA | 3.5mA | 12mA | 140AH |
| Standby | 500nA | 10uA | 5uA | |

*Figure 3. The main parameters of components*

Figure 3 shows the main features of components. The total Device length: 3.9 + 3 + 5 =11.9 mm, Width is <=10 mm. and Thickness(PCB + chip) is less 5 mm. So the space inside the shell is enough for the data logger with dimension W 15 mm x L 20 mm x H 15 mm( PCB with all chips, capacitors and resistors and battery).

## 2.3 Implementation and prototype

The data logger has been designed and implemented as simple and small as possible by choosing small chips and battery and other components. The final design version contains only three IC chips: MEMS MotionTracking device MPU-9250, 128Mb serial flash memory S25LP128 and microcontroller PIC16LF1825 plus a coin battery CR1632. All of three chips have SPI (Serial Peripheral Interface) available and are used to

communicate among them. The 2 pins of the microprocessor are for power supply and 3 pins for SPI connected to IS25LP128, 1 pin for acceleration data ready interrupt and 2 pins for i2C connected to MPU-9250 as shown in Fig 4.

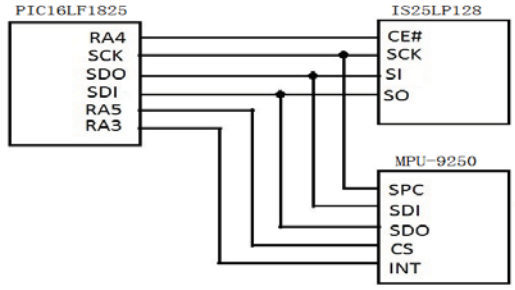



*Figure 4. System Architecture of Data Logger.*

Figure 5 shows the prototype built with MPU-9250 breakout and PIC1LF1825 and the IS25LP128 chips.

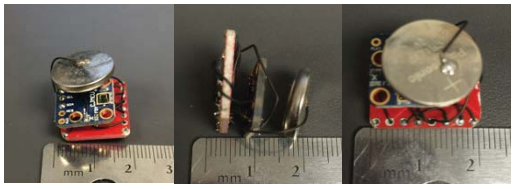


*Figure 5. DataLogger, The prototype with MPU-9250 breakout and CR1632 battery*

The DataLogger has three layers:

1. Battery(CR1632 140mAh)
2. MPU-9250 Breakout(deep blue)(15 mm x15 mm)
3. MCU and IS25LP128 on a small red PCB(15 mm x 20 mm)

# 3 Software design for MCU

The circuit design meeting the requirements is built on a prototype PCB board and is illustrated in last section. In order to make it work for our specific application. We need to write and program software into MCU→ PIC16F1825 chip. The data logger software was developed by using Microchip's MPLAB X IDE with XC8 compiler and programed into PIC16F1825 by using Microchip's Real ICE which is a develop device with programming PIC16F1825 and debug function.

## 3.1 Software functions

The software functions of the PIC16LF1825 are :

1. Configure PIC16F1825's SPI bus at pin 5,6 and 7, interrupt function at pin 4 , SPI slave selection pin: pin 1 for MPU9250 and pin2 for IS25LP128 and clock rate at 4 MHZ;
2. Load DMP driver into MPU-9250 and configure MPU-9250 Accelerometer scale rang as {-4g, 4g}, Gyroscope as 500 degree/,S sampling rate as 100/S and data ready signal output at "int1" pin;
3. Enable IS25LP128 page program;
4. Waiting for the data ready signal from MPU-9250;
5. If the data is ready, PIC16LF1825 will response and start reading from MPU-9250;
6. Write them into IS25LP128 and return to 4.

Figure 6 shows entire software flow chart.

The sampling rate of MPU-9250 is set as 100 times per second, each time there are 20 bytes (8 bytes Quaternions , 6 bytes Angular rates and 6 bytes Accelerations) data generated, that means there are less than 10ms for interrupt service routine to read, write data and return.

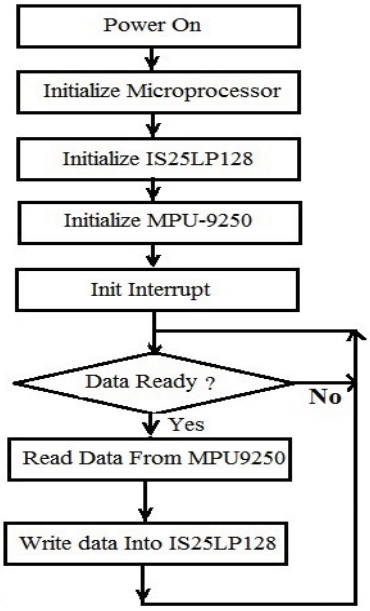



*Figure 6. Software flow chart.*

Lab tests show that the data ready interrupt service routine totally takes about 7.6ms to complete its task.

Recorded data(Quaternions ,Angular rates and Accelerations) will be read out from chip using serial flash memory programmer in binary format and write as a binary file on PC. Further data processing will be discussed in next section.

How many hours the SPI flash memory can store? Based above discussion, there are 1200 bytes per second (128Mb/8)/1200→7800 seconds, that is 2

hours and 10 minutes data can be stored in IS25LP128 chip.

### 3.2 Configuration and SPI switching

PIC18LF1825 has a MSSP model with configurable SPI/I2C bus. The MSSP module can operate in one of two modes:

• Serial Peripheral Interface (SPI)

• Inter-Integrated Circuit (I2C™)

First the MSSP need to be configured work as SPI.

SPI has 4 Modes, MPU-9250'SPI works as Mode 0 and IS25LP128'SPI works as Mode 3, thus in interrupt service routine (ISR function), MSSP need to be configured as SPI mode 0 for MPU-9250 to read measured into RAM, and use mode 3 for MPU-9250 for IS25LP128 to write the measured data into it.

## 4 Data Processing

The data processing software was developed to transforms the binary data file into a text file on PC, computes the earth Gravity, linear acceleration, rotate the linear accelerations from body frame to world frame. Then the linear accelerations in world frame are filtered and integrated twice to get the position in X, Y and Z directions. The path of data logger traveled can be visualized on the screen of PC except the readable data file.

### 4.1 Data transformation

Recorded raw data is in signed acceleration integer binary in SPI serial memory are in Gravity unit and in body frame, will be read out by using SF600 SPI memory programmer with clip and stored into a binary file on PC compatible computer. We first transform the acceleration data based on the configured scale{-4G, 4G} ( 1 G = 8192.0 according to the 16-bit ADC of MPU-9250 ), thus

Thus Acceleration=RecordedBinary/8192.0

Then we transform the quaternion:

q= RecordedQuqternion/16384.0

from Gravity unit to M/S^2 unit and write them into a file in decimal format.

### 4.1 Compute Gravity, accelerations and estimating the position/path with strapdown inertial navigation algorithm

Gravity refers to the earth gravity excluding the acceleration caused by the user, three dimensional vector consisting of a relative angle between the device body-frame and gravity vector.

Gravity will be computed based on Quaternions, using following formula:

```
g[0] = 2 * (q[1]*q[3] - q[0]*q[2]);
g[1] = 2 * (q[0]*q[1] + q[2]*q[3]);
g[2]=q[0]*q[0]-q[1]*q[1]-q[2]*q[2]+q[3]*q[3];
```

where g is Gravity, q is Quaternion.

Linear accelerations are calculated by eliminated Gravity from raw accelerations. We rotate the linear accelerations from body frame to world frame based on Quaternions, filter out the noise and computing velocity and distance in X, Y and Z direction by integrating rotated linear accelerations once and twice respectively. Figure 7 shows Orientation of Axes of Sensitivity and Polarity for Accelerometer and Gyroscope(body frame) and World Frame (ENU coordinates) .

Figure 8 shows the entire procedure of data processing

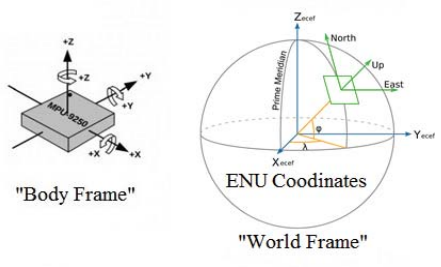



*Figure 7. Body Frame and World Frame (ENU coordinates)*

Figure 8 shows the Data process procedure.

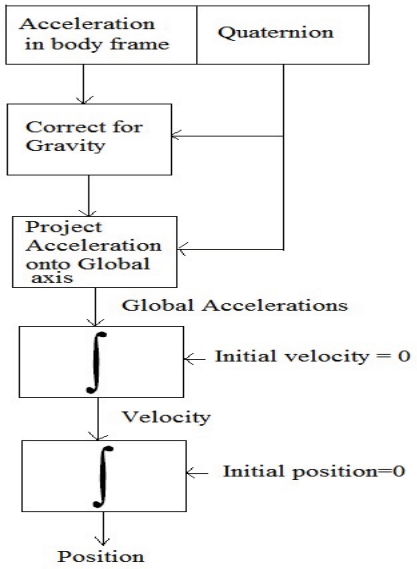



*Figure 8. Data processing procedure (Strapdown inertial navigation algorithm)( modified based on [5])*

The body frame is different from ENU (World/Global Frame), because its orientation may have been changing during testing. All measured values are based on its body frame. To compute the drilling position/path, the measured acceleration must be converted from body frame to EMU coordinate. The ENU coordinate system origin is set at the end of drilled lateral. Thus when the test starts, the DataLogger is pulled out from the end {0,0,0} of lateral to wellbore.

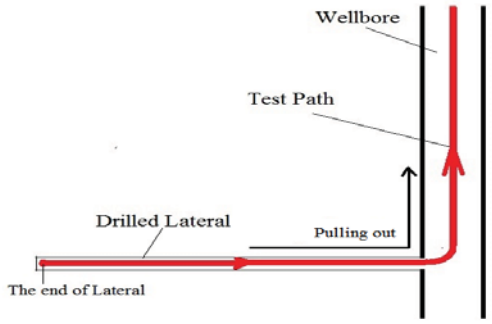



*Figure 9. Datalogger test(or moving) path(red) in lateral and wellbore as pulling out*

The measured accelerations are based on body frame, we calculate Gravity based on Quaternion, then the linear accelerations are computed by eliminating gravity from Accelerations, then based the Gyroscope and accelerations, we use Quaternion to rotate the linear accelerations from body frame to world frame.

The configured scale MPU-9250 in the data logger is {-4G, 4G}. Though MPU-9250 provides 16-bit data output, i.e. the raw data is signed 16-bit integer in Gravity unit. Raw data (signed integer in G unit) can be transferred to signed decimal float in M/S^2 unit with following formula:

Meter_Second2=Gravity*4/(2^16)*Raw
=0.000299377*Raw

Where "Raw" is raw acceleration data in 16 bit signed binary integer.

"Gravity" is 9.81M/S^2.

After the transformation, filter is applied to the transferred data, then we integrate filtered acceleration data once to get velocity and again integrate velocity to get the position/distance. Following is the codes:

```
for(i=0;i<Filtered_Data_Length-1;i++)
{//First integration---->computing velosicy
V[i+1][0]=V[i][0]+(A[i][0]+A[i+1][0])* 0.5*Delta_T;
V[i+1][1]=V[i][1]+(A[i][1]+A[i+1][1]) *0.5*Delta_T;
V[i+1][2]=V[i][2]+(A[i][2]+A[i+1][2]) *0.5*Delta_T;
   //Second integration---->computing position
P[i+1][0]=P[i][0]+(V[i][0]+V[i+1][0]) *0.5*Delta_T;
P[i+1][1]=P[i][1]+(V[i][1]+V[i+1][1]) *0.5*Delta_T;
P[i+1][2]=P[i][2]+(V[i][2]+V[i+1][2])*0.5*Delta_T; }
```

# 5 Test results

After implementing prototypes, we ran two stages test:

1. Lab test to make sure the dataLogger can properly work and the data processing software (algorithms) can properly compute(estimate) the device position with an error less than 10%.
2. Field test to calibrate the DataLogger in the real world and estimate the positions of drilled laterals.

## 5.1 Lab Test

After the prototype was built and software was developed , we programed the codes into PIC16F1825 and ran Lab tests to make sure the DataLogger can work properly as designed and data processing tool can compute the data correctly. Figure 10 shows the raw data (blue) and Filtered data (red) in X-axis processed by data processing software.

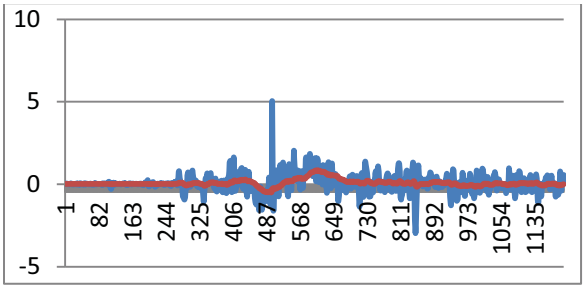



*Figure 10. Raw data (blue) and filtered data (red) in X-axis*

Calculation and test show that the cell battery (LR1632 140mAH) can power the data logger to work more than 2 hours and the flash memory (IS25LP128) can store 2 hours and 10 minutes data. The calculated velocity and position could be calculated with an error within +/-10% when it moved 30 feet on floor.

## 5.2 Field Test

In oil field the entire procedure of the drilling path test will no more than 40 minutes from power on data logger and installing the data logger to finishing the test and power off the data logger.

In order to compute the length of a lateral, it must be known when the DataLogger enters the shoe, when it turns into the lateral, when it reaches the end of the lateral and each path component as it is pulled back out. To easily identify the each position, a test procedure was established. This procedure is illustrated on a recording of raw gravity curves in Figure 9. Figure 10 shows a zoomed in version of

the data in Figure 9. The data acquisition process is as follows:

1. Power on with 30 minutes delay before recording;
2. Trip down well, sensor start to record at 30 minutes;
3. Stop at 30 feet above the shoe for 2 minutes;
4. Enter shoe and reach the end of lateral;
5. Stop and wait 2 minutes;
6. Pull out through lateral and cross shoe;
7. Stop at 30 feet above shoe for 2 minutes;
8. Repeat process if multiple records are desired, or;
9. Pull DataLogger out of the well;
10. Power off and unload recorded data into a laptop;

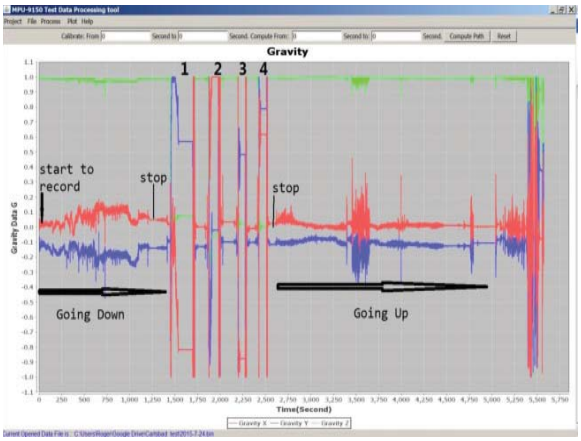



*Figure 11. A full record of a sensor run that includes four trips in and out of the lateral. The green track is the Z-axis of the accelerometer, the red and blue are X and Y axes respectively. From left to right, record begins during the trip down the well, stops above the shoe, records 4 measurements in and out of the lateral, stops above the shoe, and is pulled out of the well.*

Since the accelerometer's Z axis is pointed to the earth's center, several features may be discerned from Figures 11-12:

1. When the Z measurement is near 1, the DataLogger is vertical
2. When the Z measurement is near zero. the DataLogger is horizontal
3. The Z measurement changes from 1 to 0 means the sensor is turning into the shoe, from 0 to 1 means the sensor is coming out shoe;
4. Variable G in X,Y and Z Axis means the DataLogger is moving;
5. Constant G means the DataLogger is at rest.


*Figure 12. Zoomed portion focusing on the fourth test run in figure 11. It can clearly be seen when the sensor turns from vertical to horizontal and when the sensor is stationary or moving.*

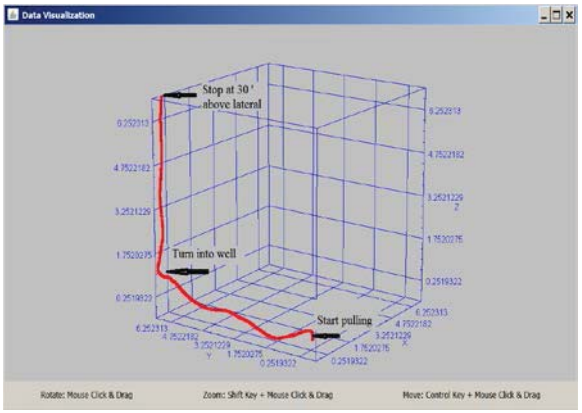



*Figure 13. Visualization of the computed track of fourth test run using data shown in Figure 11.*

In total for the fourth test run, from time: 2509.0 seconds to 2545.0 seconds, 26 seconds of data were recorded during which time the DataLogger traveled through the lateral horizontally (31.5 ft) then turned into the shoe, and was pulled out vertically, stopping at 30 ft above the lateral. Tabulated data for Barbie #3 tests, and Federal 00 #3 tests are shown in Tables 1 and 2 at last page, and the track of the mapped lateral is shown in Figure 13. The best results were from test runs recorded with a pull speed of about 0.8 -0.9 ft/s for the shorter Barbie #3 laterals, and between 1.25 and 1.33 ft/s for longer laterals in the Federal 00 #3, these numbers are in bold in Tables 1 and 2. Best results were towards the end of the fieldwork, in part due to adjustments to software and testing protocols as information was acquired, processed, and optimized throughout the test period.

# 6 Conclusions

A small, on-system DataLogger was developed to measure the laterals post-drilling, to allay a frequent concern of small operators likely to use the technology, "How do I know where it went?" The DataLogger is small (20 mm D × 15mm in L) and can record 3-axis gyro and 3-axis acceleration for 2+ hours.

A user-friendly data processing tool was developed, which includes data transformation, algorithms and filters for computing path length of the emplaced laterals, and to provide data visualization.

Field-testing has proven that the DataLogger can work, and errors on longer laterals, measured at an appropriate pull speed can be on the order of 10%.

# 7 Acknowledgements

Funding for this project is provided by RPSEA through the "Ultra-Deepwater and Unconventional Natural Gas and Other Petroleum Resources" program authorized by the U.S. Energy Policy Act of 2005. RPSEA (www.rpsea.org) is a nonprofit corporation whose mission is to provide a stewardship role in ensuring the focused research, development and deployment of safe and environmentally responsible technology that can effectively deliver hydrocarbons from domestic resources to the citizens of the United States. RPSEA, operating as a consortium of premier U.S. energy research universities, industry, and independent research organizations, manages the program under a contract with the U.S. Department of Energy's National Energy Technology Laboratory.

# 8 Future work

To improve the Datalogger:

1. Add the compass to use 9-axis fusion solution, the steel shell needs to be replaced with a shell made of low magnetic permeability material.
2. Use Embedded MotionDriver v6.1 (9-axis fusion solution), PIC16LF1825 needs to be replaced with a MCU with larger program memory.
3. Optimize the algorithms and filters to improve the accurate of track estimating.

# 9 References

[1] InvenSense Inc., "MPU-9250 Product Specification Revision 1.0", 2014

[2] Integrated Silicon Solution Inc."IS25LP128 128M-BIT 3V Serial Flash Memory datasheet", 10/3/2014

[3] InvenSense Inc., "Embeded MotionDriver v5.1.2",2012

[4] Micochip Technology Inc, "MPLAB X IDE ", 2012.

[5] Oliver J. Woodman, "An introduction to inertial navigation", 08/2007

[6] Micochip Technology Inc, "XC8 programming ", 2014.

[7] Micochip Technology Inc, "PIC16F1825 data sheet," 2012,pp.1-5

[8] Andrew Blak etc, "Deriving Displacement from a 3 axis Accelerometer", 2012.

[9] SPI™ Overview and Use of the PICmicro Serial Perip Interface
http://ww1.microchip.com/downloads/en/devicedoc/spi.pdf

Table 1. Calculated Lengths and Speeds for the Five Laterals Tested in May-June, 2015 at the Barbie #3

| Test Date | Actual Length (ft) | Test 1 - Predicted Length(ft)/Speed (ft/s) | Test 2 - Predicted Length(ft) /Speed (ft/s) | Test 3 - Predicted Length (ft) /Speed (ft/s) | Test 4 - Predicted Length (ft) /Speed (ft/s) |
|---|---|---|---|---|---|
| 5/19 | 14.50 | 16.47 / 0.71 | **15.03 / 0.88** | | |
| 5/24 | 14.50 | 9.83 / 0.66 | 6.93 / 0.90 | 9.83 / 1.0 | |
| 5/29 | 14.50 | **12.37 /0.81** | 11.77 / 0.88 | 11.67 / 0.66 | 9.13 / 0.97 |
| 6/1 | 14.50 | **15.1 / 0.81** | 12.33 / 1.00 | 19.1 / 0.92 | |
| 6/3 | 14.50 | 7.43 / 1.23 | | | |

Table 2. Calculated Lengths and Speeds for the Five Laterals Tested July–October, 2015 in the Federal 00#3

| Test Date | Actual Length (ft) | Test 1 - Predicted Length(ft)/Speed (ft/s) | Test 2 - Predicted Length (ft) /Speed (ft/s) | Test 3 - Predicted Length (ft) /Speed (ft/s) | Test 4 – Predicted Length (ft) /Speed (ft/s) |
|---|---|---|---|---|---|
| 7/24 | 31.50 | 37.40 / 1.19 | **31.90 / 1.33** | 21.43 / 2.1 | 23.43 / 2.62 |
| 10/8 | 30.00 | 32.47 / 1.20 | 31.67 / 1.18 | 25.60 / 1.13 | |
| 10/9 | 30.00 | **28.83 / 1.29** | 32.73 / 1.09 | 26.80 / 1.20 | |
| 10/12 | 30.00 | **32.17 / 1.29** | **30.40 / 1.27** | **33.40 / 1.27** | |
| 10/16 | 30.00 | 30.60/1.18 | **33.53/1.25** | **33.43/1.25** | |

# FPGA based Networked Embedded Systems Design and Prototyping: Automobile oriented Applications

B.Senouci, R.Zitouni

Central Electronic Engineering School, ECE-Paris, France
Embedded Systems Department
LACSC Laboratory
37 Quai de Grenelle, 75015 Paris
senouci@ece.fr

**Abstract —In 1968 Volkswagen integrates an electronic circuit as a new control fuel injection system, called "Little Black Box". It's considered as the first embedded system in the automotive industry. Presently, every year, automobile constructors integrate new embedded systems into their vehicles. Behind these automobile's embedded systems, a sophisticated HW/SW architecture, which is based on Multiple CPU is built. In this paper we describe our experience within a progressing work in using FPGA (Field Programmable Gate Array) to design and prototype wireless networked embedded systems. We built a hardware platform based on FPGAs in order to emulate a wireless networked embedded systems for V2X automobile applications. Also, we discuss the emulation of on-car embedded systems that communicate and exchange data wirelessly. Firstly we show how a HW platform based FPGAs can be a very attractive approach; secondly, we describe a real experience in building a wireless networked hardware platform based on Zynq 7020, finally, we share our preliminary results by discussing our choices.**

***Keywords***: *Multiprocessor Embedded Systems, Embedded OS, FPGA, HW/SW Architecture, Wireless Sensor Networks, V2X Communications*

## 1. INTRODUCTION

In order to replace the duel carburetors Fastback and Squareback system that control the fuel injection, in 1968 Volkswagen 1600 integrates an electronic circuit (more than 200 transistors, resistors, diodes and capacities) as a new control fuel injection system, called "Little Black Box" [1]. Then, the first embedded system for automotive industry is born. Presently, every year, automobile constructors integrate new embedded systems into their vehicles. On one hand, the massive usage and availability of these embedded devices on the marketplace bring products to a price consumers can pay for, on the other hand scaling down of semiconductor technology below 22 nm, will surely reach many of these devices, by improving their application's diversity and availability in automobile industry.

These tiny devices integrated in automobiles collect and exchange information to control, optimize, and monitor many of the functions that just a few years ago were purely mechanical.

The technological advancements of embedded system and electronics within the vehicle are being driven by the challenge to make the vehicle safer, smarter, more energy efficient, autonomous, and networked. ASIC/ASIP based embedded CPUs, from on-chip system to FPGA, are the command center for these embedded systems design.

Automobile applications take a real advantage from this embedded integration advance. Every year, automobile manufacturers pack new embedded system into their vehicles. Currently, 100% of new car production integrates at least one embedded device. This number is being increased exponentially. Table I shows several examples of current embedded systems for automobile.

| Embedded Systems in Vehicles | Application Example | Safety |
|---|---|---|
| Engine Control Systems | Fuel Injection Control | + |
| Speed Control Systems | Cooperative Adaptive Cruise Control, ABS System | + + + + |
| Driver and Car Safety systems | Immobilizer Unit, proximity Alert | + + + |
| Seat and Pedal Controls | SRC (Sanitary Remote-Controlled) | + + + |
| Route and traffic monitors | Navigation Systems | ++ |
| Voice Activation and commands | Car Lock | ++ |

**Table 1: embedded systems for automobile**

On the other hand, a new direction in short-range wireless applications for few meters distances has appeared in the form of high-speed data transmitter/receiver devices [2]. The vision with short range wireless communication can be implemented within any embedded system device and thus they are the focus of this paper. This is being developed by embedded systems leaders to develop the new Internet of Things era; connected vehicles illustrate the typical example.

Unlike general automobile embedded system designs, where most researches are involved in the software design part, and used pre-designed IPs for the hardware components. In the new smart embedded systems for automobile (ADAS: Advanced Driver Assistant System) [3] a new design flow has to be proposed. In addition to the traditional embedded systems constraints (performance, time to market, size, SW memory foot-print…). New constraints (Safety, Smartness of the systems, dependability, networks …) have to be taken in consideration in the new design flow, and at a high level of integrity. Furthermore, software will be implemented as a system with several layers (Application,

control, communication…). Instead, current HW/SW methods design is mainly based on a software environment simulation tools, within a functional validation at high level of abstraction. Thus, this method doesn't provide an overall and effective evaluation for the performance and cost of entire automobile's embedded system. Besides, in this technique it's hard to simulate the wireless communication part. The development of FPGA technology brings advantages to the embedded system in size, cost and performance. It also introduces the capability of intelligence to I/O processing. Furthermore, it can solve the timing and synchronization problems, which are hardly solvable by pure software programs on Real Time Operating System (RTOS). Many FPGA vendors including Actel, Altera, QuickLogic, Xilinx, have all provided both hard and soft cores embedded in the latest FPGA silicon chips. Programmable FPGA based design is being the mainstream of embedded system designers. And it makes it easy the embedded HW/SW co-design which is necessary in order to benefit from the advantages of both worlds (SW & HW).

In our work, we investigate this flexibility of FPGAs in order to design a Networked Embedded System (NES). This NES emulate automobile networks that communicate wirelessly with each other for a safety purposes. We investigate a hardware platform based design approach [13] using several FPGA instances to design and validate Networked Embedded System (NES). This NES emulate automobile networks that communicate wirelessly with each other for a safety purposes for example. All is become possible by the amazing capacity of integration, then we can embed several electronics systems that make the cars network smarter to estimate and manage any new danger situation. As we said previously, most of the ongoing research projects on wireless communications network are interested in short range radio for data exchange, smart cars concept takes a real advantage of these research projects to increase its efficiency and performance. There exists a very large amount of wireless technologies today. Table 2 shows different examples of short range wireless technologies and their applications.

**Table 2: Short Range Wireless Technologies**

| *Technology* | *Frequencies MHz* | *Applications* |
|---|---|---|
| **802.11p, WAVE** | 2,4 GHz | Vehicle Communication |
| **Bluetooth** | 2,4 GHz | Audio Applications |
| **IEEE 802.15.4** | 2,4 GHz | Wireless Networks |
| **RFID** | 902 to 928 GHz | Video Transfer |
| **ZigBee** | 2,4 GHz | Home, Monitoring & control |

On the other hand, the safe and the reliable communications between automobiles can't take for guarantee; as a correctness communicated data may save many lives, a miss or wrong one may cause a real damage.

These electronics systems integrated in today's cars has to be designed, validated, and prototyped at high level of integrity. In networked embedded systems for automobile applications, we distinguish two network communication schemes:

1. In-Car Networked Embedded System: Here the embedded devices integrated in the car connected wired via a local network (ex: CAN) [14]. These embedded systems are controlled by a high level supervisor which may considered as an operating system that controls the whole automobile electronics systems.
2. Car-to-Car Networked Embedded System: The second communication scheme is wireless based; it manages the communication between automobiles in order to exchange data about the environment where both automobiles operate. This communication is our concern in this work.

These embedded wireless networked nodes are deployed in automobile applications as ADAS to collect information data related to automobile's environment and communicate it around to avoid harsh situations.

The remainder of this paper is organized as follows. Section 2 presents some approaches and related works. Section 3 focuses on the challenges presented by the embedded systems design for automobiles. In section 4 we give details about our networked FPGA platform concerning hardware and software architectures. The last section discusses implementation details. The paper ends with a conclusion and some future work.

## 2. APPROACHES AND RELATED WORKS

Advances in integrated circuit and embedded technology design, make possible to deploy heterogeneous components in embedded design and networks. Many research projects/groups are exploring the issues related to the design of these networked embedded systems for deployment in automobile applications.

In overall, most of the techniques used for embedded system design are based on simulation practice, using software components models. In such techniques many hypotheses on the system are made early in the design process, especially for the hardware architecture, and have to be verified in a tardy stage of the design process (prototyping phase). If one or more of the assumptions is incorrect, it outcome a significant re-design and debug overtime as compared to the initial expected design time. Another ignored factor in automobile business, there is a risk that these shortcomings will be missed the time-to-market of the product, and lead to displeased customers. Simulation-based techniques raise at least two inconveniences: 1) A time consuming technique (very slow); 2) Very hard to enable a simulation in the case of wireless components (e.g.: software simulation model for sensor)

Furthermore, the general state-of-the-art methods that study, and analyze wireless communication and FPGA based design are various. Several works are reported in these research topics. There exist several different systems that implement embedded solution for automobile applications. The purpose of this section is to give an overview of some of them.

Since long time the safety in automobile industry is considered as the highest priority subject, to develop and provide smart systems that helps drivers to avoid road's accidents. Advanced Driver Assistance System (ADAS) is introduced specially to deal with this concern and proposes solutions with the intention of assist the car's drivers with an efficient warning. Many ADAS systems are available on the market place and already integrated in cars. Parking aid with its ultrasonic sensors or embedded cameras in the new cars generation illustrates existent examples of these systems. Mainly two categories of ADAS are available, radar based or camera-vision based.

Mody and al [5], introduce a Front Camera (FC) based ADAS system uses computer vision based techniques to detect obstacles (e.g. pedestrian, cyclist etc) on road from captured image. Typical Image Signal Processing (ISP) is developed to cater cellphones and Digital Still Cameras (DSC) for purpose of human viewing unlike computer vision algorithms whose purpose is to help driver. In [4] authors present the design and implementation of modular customizable event-driven architecture with parallel execution capability for the first time with wireless sensor nodes using stand-alone FPGA. Authors in [6] present a new FPGA based framework for dependability study of wireless short range radio applications. In this framework, authors built a platform using SmartFusion FPGA instances. And then, define a methodology to investigate the dependability of short range application at three main levels, analog, digital, and wireless. EyeQ2 [7] systems is one example of a single chip dedicated to automotive security applications using vision system, that consists of two 64- bit floating-point RISC 34KMIPS processors for scheduling and controlling the concurrent tasks, five vision computing engines and three vector microcode processors. This architecture provides support for a specific set of real time data intensive applications. In [8] authors present a dynamically reconfigurable MPSoC (Multiple Processor System on Chip) prototype for AutoVision system; It offers functions such as object edge detection or luminance segmentation, and are implemented as dedicated hardware accelerators to ensure real time processing. Mainly, these ADAS based vision systems suffer from their reliability, particularly in harsh environment where the visibility become difficult or even impossible (cloudy weather, rain and snow…etc).

In [9] and [10] authors present a radar based Embedded MTT-ADAS Application (Multiple Targets Tracking) ADAS. This application is prototyped on the top of an Virtex 5 FPGA fabric using an open source FreeRTOS (Real Time Operating Systems).

## 3. Embedded Systems for Automobile: Challenges

Behind these automobile's smart embedded systems, an heterogeneous HW/SW architecture is built, composed principally of, digital components, analog components, processing elements, and Tv/Rv communication modules. Figure 1 shows a typical integrated embedded system for automobile. As far as the wireless communication in networked embedded systems is concerned it is desired that a data must be received at the final destination. This data may represent vital information for an emergency requirement in order to avoid accidents or just to assist drivers in hard driving situations, by providing him real-time information about the environment where he is growth in.

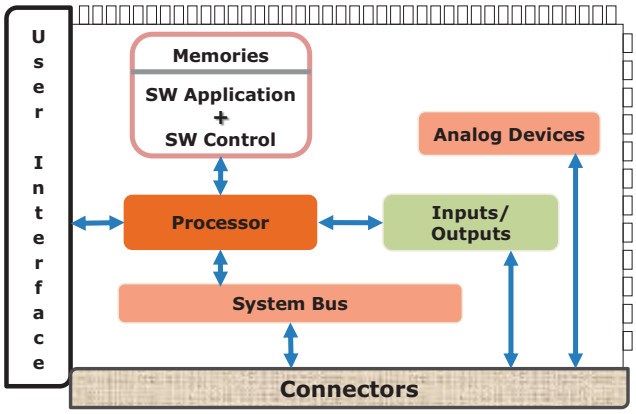



**Figure 1: Automobile Embedded System Architecture**

Networked embedded systems wirelessly linked for automobile applications require a sophisticated HW/SW architecture with a complete design methodology, and then present several challenges:

- Different software levels are needed to supervise these systems embedded in the automobile; the specification and the design of these software levels is one of the hardness tasks in the design flow.
- Also, one of the vital software level is the smart application level, then defining a smart algorithm mainly based on machine learning and deep learning become a real challenge,
- Behind the V2X communication in automobile applications, an embedded system based on heterogeneous components is built (analog, digital, wireless…), then how can we define the different steps of its design, taking into account the complex tradeoffs between safety, performance, cost, and time to market?
- The application software has to be designed taking into account more and more smartness within an efficient software algorithm.
- Reliability and safety attributes in automobile's embedded systems are a key matters. Then, how they can be approved besides the pure functionality, and the correctness of the exchanged data In-car and Car-to-Cart networked embedded systems.
- On the hardware side, specifying the architecture in order to improve the adequacy algorithm/architecture and get the determined

performance in our system become a hard task that need a deep design space exploration.

- The processing efficiency and flexibility can be compensated with the use of a reconfigurable FPGAs, then how can we imagine a NES using FPGA based design approach?

One of the major challenges addressed in this paper is to investigate the use of FPGA device for automobile application design and emulation. In application for automobile, the embedded computing are purely software and mainly based on embedded processor, which is having limitation in terms of flexibility, prototyping, and computational capability related to various applications. Using FPGAs instances make easy the HW/SW co-design in order to get the benefits from both worlds (software and hardware).

Then, we share our experiences in this progressing work by describing the selection, specification and realization of a hardware platform using Zynq FPGA instances, for an early design and prototyping of automobile embedded application. We first present a hardware platform-based design approach as the most attractive solution for embedded automobile application design. Unlike the simulation-based approach, the platform-based framework allows to design and analyze the networked embedded system in a real environment with direct real-time execution at the desired cycle accurate level.

## 4. NETWORKED EMBEDDED SYSTEM PLATFORM

We have used the Zynq FPGA and the TelOSB devices as a COTS (Commercial Off-The-Shelf) in order to emulate a networked embedded architecture for automobile applications. The data is transmitted from one Mode (Mobile Node) to another one. Each Mode is composed with a TelOSB connected to an FPGA Zynq.

In Figure 2 we show the global view of the FPGA based platform for networked embedded system design [3]. Each FPGA may represent one or more embedded system, and then emulate an automobile's embedded system. These embedded nodes communicate with each other according to a standardized network topology and may be reconfigurable at some level. In our design methodology, a V2X communications shim is implemented via TelOSB modules and makes references mainly to V2V (Vehicle to Vehicle) communications. These wireless communications is considered one of the key features in the context of designing embedded systems for automobile applications.

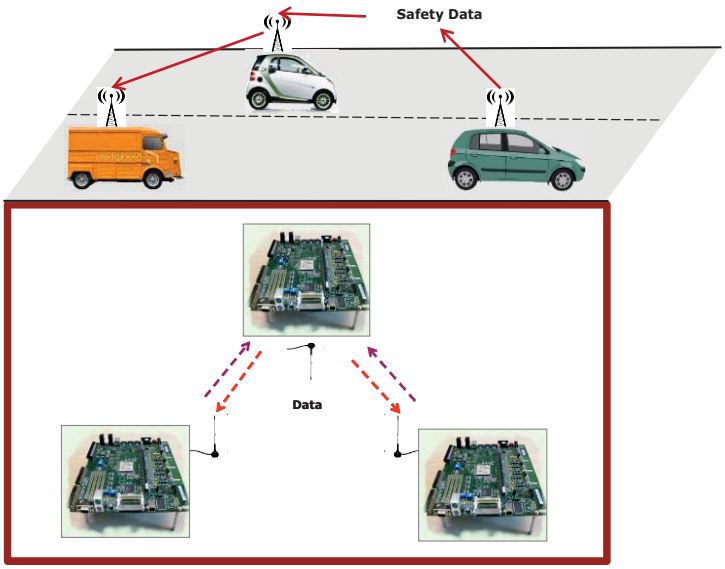



**Figure 2: Global view of the Hardware platform**

### 4.1 Hardware architecture overview

We are using a Xilinx Zynq-zc702 platform [11] and TelOSB sensor for our platform building presented below. The Zynq board is based on the Zynq ZC7020 device. This component regroups large reconfigurable logic, communication interface controllers, and one processing unit that integrate two ARM Cortex-A9 CPUs. Each CPU has its own cache memory banks for instruction and data, and one common level 2 caches memory (Figure 3-2). The clock frequency of each unit is 667 MHz. The communications between the interface controllers and the CPUs (as well as HW IP), are performed via AMBA/AXI bus network. Figure 3-1 shows the internal architecture based on two processors ARM, local and global memory system, and the AMBA/AXI bus. Figure 3-1, shows the experimental setup, with the TelOSB wireless node connected to the embedded Zynq architecture via USB hub.

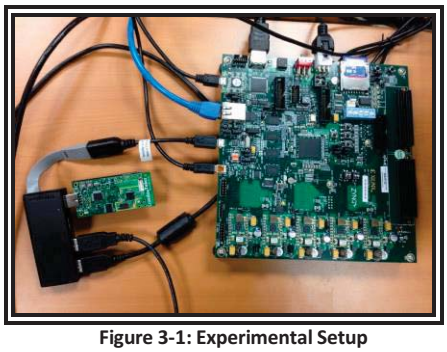


**Figure 3-1: Experimental Setup**

The board proposes a large panel of interfaces [11]. In our NES wireless platform, an Ubuntu/Linaro compiled and ported on each FPGA instance. The Linaro is booted from an SD card memory [12]. On the top of the linaro a customized Contiki operating systems is used in order to get the data from the TelOSB COTS component, connected to the system via an USB hub. The visualization of results produced by the

system is performed via HDMI interface. The SD card contains the boot and the OS files. After the initialization phase, the OS uses available DDR3 banks as a memory space. An additional keyboard and mouse are included via an USB hub to our final system configuration.

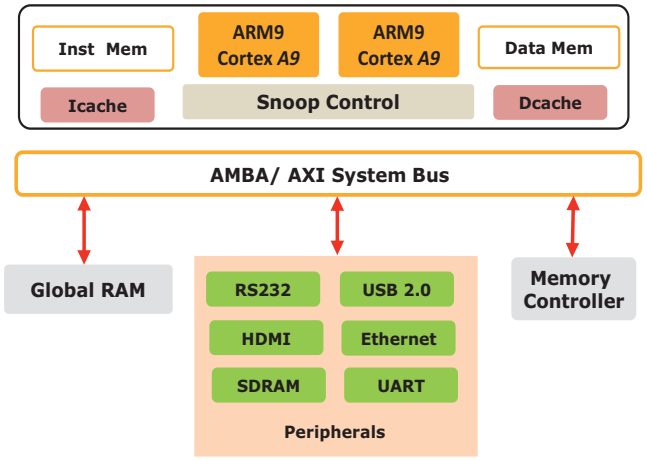



Figure 3-2: Zynq based Hardware Architecture

### 4.2 Software architecture Layers

Figure 4 shows the different layers of the software architecture. In order to master the complexity of the embedded software, we divided it into several parts. Three main layers are identified: the application layer, the embedded operating system layers and the hardware abstraction layer. In the application layer we find the software tasks that allow the communications managed by the ContikiOS layer.

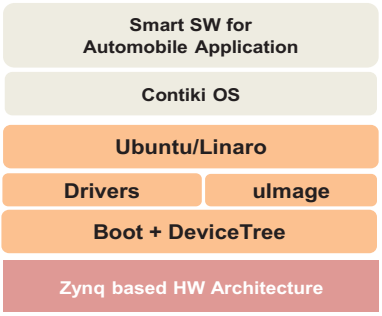



Figure 4: Software Architecture Layers

While the Embedded OS layer acts as an adaptation layer and manage the execution of the application tasks on the Zynq based architecture. The hardware abstraction layer recognized here by the drivers, the boot and the DeviceTree give an insight of the target hardware. The first layer of the operating system is Ubuntu/Linaro based; it was ported on the ZC702 FPGA [11], [12]. Then, on the top of the Linaro we compile a version of Contiki to manage the wireless data package exchange. As primary results, in Table 3 we show the memory footprint of the Operating systems. This was the result of compiling the "C" source files using ARM processor cross compiler, with –Os (size optimized) as optimization option.

Table 3: Code size of the software parts

| Embedded OS | Kernel | Boot | Device Tree |
|---|---|---|---|
| Linux-Linaro | 3.09 M-Bytes | 4.16 M-Bytes | 10 K-Bytes |
| Contiki OS | 40 K-Bytes | 2 K-Bytes | - |

### 4.3 Hardware Architecture

On the HW side, the logical synthesis process is performed using XST tool. It does take about half an hour to completely synthesis the hardware architecture on the Zynq zc702 platform. Table 4 shows the synthesis logic results; where the components names refer to the HW components mentioned in Figure 3-2. The other columns represent the size in term of logic gates/Look-Up-Table, memory banks (BRAM) and Flip-flops.

Table 4: Synthesis results of the hardware part

| Components | LUTs Used | Flip Flop Used | BRAMs |
|---|---|---|---|
| All System | 34568 | 36004 | 10 |
| System Processing (Two ARM9) | 349 | 81 | 0 |
| System Bus | 6658 | 8251 | 4 |

## 5. Conclusion:

This paper deals with the design and prototyping of embedded systems in automobile applications. Here, a first step from a progressing work is presented. For this purpose, we firstly and successfully built a hardware wireless platform. The platform is built using several instances of one of the FPGA platforms (Zynq 7020) and COTS (TelOSB) for wireless links. The goal behind the construction of the wireless networked hardware platform is to design, validate, and prototype a wireless automobile's NES (Networked Embedded Systems) in an environment very close to the real application settings. This FPGA based NES platform present an improved alternative for the time consuming simulation technique in embedded systems design. Based on this platform a new design flow/methodology with a fast design space exploration is being developed for smart embedded automobile applications. In this networked hardware platform, the combination of programmable logic, wireless COTS, and IP cores including embedded processors can speed up the automobile applications design and realize the hardware programming. Within the proposed networked FPGA platform, and the advance in the appropriate design flow the trend is believed to show greater potentials.

## Acknowledgment:

We would like to express our gratitude to the individuals, companies, and our academic partnerships we work with on the "ASSIA" collaborative project.

## 6. REFERENCES


[1] "How VW Fuel Injector Works: A mini-Computer Aids Economy, Cuts Pollution" Chicago Tribune, Sunday, February 25, 1968

[2] Alan Bensky "Short-range wireless communication: Fundamental of RF system Design and Application" 2005.

[3] Seminar Link "Embedded System in Automobiles" www.seminarlinks.blogspot.com

[4] Liao et al. "FPGA based wireless sensor node with customizable event-driven architecture" EURASIP Journal on Embedded Systems 2013, 2013:5 http://jes.eurasipjournals.com/content/2013/1/5

[5] Mihir Mody and Al. "Image Signal Processing for Front Camera based Automated Driver Assistance System"

[6] Senouci, B., et al. "Dependability Investigation of Wireless Short Range Embedded Systems: Hardware Platform Oriented Approach." Journal of Embedded Systems 3.1 (2015): 1-10

[7] G. P. Stein, E. Rushinek, G. Hayun, and A. Shashua, "A computer vision system on a chip: a case study from the automotive domain," IEEE Conference on Computer Vision and Pattern Recognition (CVPRW'05), p. 130, June 2005.

[8] C.Claus, W. Stechele, and A. Herkersdorf, "Autovision– a run-time reconfigurable mpsoc architecture for future driver assistance systems," Information Technology, vol. 49, no. 3, pp. 181–187, 2007

[9] B.Senouci, S.Niar, Julien Dubois, Johel Miteran "Embedded MTT-DAS Application Prototyping on an FPGA based Multiprocessor Architecture" International conference on Embedded Systems and Applications, ESA, July 2014, Nevada, USA

[10] J.Khan, Smail Niar, Mazen Saghir, Yassin El-Hillali, Atika Rivenq, "Driver assistance system design and its optimization for FPGA based MPSoC," sasp, pp.62-65, 2009 IEEE 7th Symposium on Application Specific Processors, 2009

[11] www.xilinx.com\zynq

[12] http://fpga.org/2013/05/24/yet-another-guide-to-running-linaro-ubuntu-desktop-on-xilinx-zynq-on-the-zedboard/

[13] B.Senouci, A.Bouchhima, F.Rousseau, F.Pétrot, A.Jerraya "Prototyping Multiprocessor System-on-Chip Applications: A Platform-Based Approach" Journal IEEE Distributed Systems Online archive Volume 8 Issue 5, May 2007

[14] CAN specification version 2.0. Robert Bosch GmbH, Stuttgart, Germany, 1991

# An Implementation of a Skin Care System for Android Phones

**Yeonbo Kim** [1], **Jihoon Baek**[2], **Byoungchul Ahn**[2]
[1] School of Electronic and Electrical Engineering, Daegu University, Gyungsan, Korea
[2]Dept. of Computer and Communication Engineering, Yeungnam University, Gyungsan, Korea

**Abstract -** *This paper presents an implementation of hardware design and software program to diagnose and analyze skin conditions with Android smartphones. This portable system has implemented on an ATmega328P microprocessor to capture skin signals and transmit them to smartphones by the earphone jack. It has sensors to measure temperature, moisture and oil level of the skin. It has UV and white LED lights to capture face images using the camera of a smartphone. The portable system can be mounted on smartphones to take skin images and adjusted the magnification using an auxiliary lens. The Android application analyzes data and images which are collected data from the skin. It monitors and stores skin conditions such as moisture, oil level, pigmentation, pores and wrinkles after processing skin images. The skin care system and its application software are tested and the experiment results show that the average accuracy of the skin care system is 96% and higher as compared with those of human examinations.*

**Keywords:** Skin Condition, Embedded, Bluetooth, Android, Diagnosis

## 1 Introduction

As smartphones have been spreading very fast, many application programs are developed and used for nowadays. Recent interests are also applied to personal health application programs for smartphones. Since there are a lot of demands for skin care to keep the beauty, personal skin care products have been introduced in the market. But it is very expensive to use them at home. According to "Beauty Organization: A Global Strategic Business Report" by Companies and Markets, they forecast to sell 600 million products of the personal beauty market in 2018[1]. The largest share of the beauty products was skin care products by sharing 55% of the beauty market in 2012[2].

It is necessary to measure and diagnose skins using smartphones and accumulate measured data for further analysis. The important skin data are moisture, pigmentation, oil, wrinkles, pores and so on. This paper presents a device design to measure and analyze skin data using Android smartphones to manage and monitor skin status personally. The hardware is designed using sensors and Bluetooth, and software design includes app-based image processing system for skin diagnosis.

This paper consists of six chapters. In Section 2, related studies are examined for skin care and measuring methods. Section 3 describes the implementation details of the hardware design, firmware and Android apps for the skin care system. And Section 4 describes skin diagnosis algorithms. Section 5 presents test results of the implemented skin care system. Finally, this paper concludes in Section 6.

## 2 Related work

There are many researches and various techniques for the skin care. There are also many expensive devices implemented skin diagnosis algorithms using cameras to measure the skin condition accurately. It is necessary to design a skin care system to measure the skin condition with low cost and to use it as a personal care device.

The skin color may vary by the ambient environment and appear as different skin color, which is referred as a skin pigmentation phenomenon. The main reason for changing the color is melanin and hemoglobin component. The skin color is determined by the amount and distribution of melanin and hemoglobin components presented in the epidermis and dermis of the skin tissue. Several studies have been conducted for pigmentation detection methods. Madasu *et al*. have proposed an extended FCCI (Co-Fuzzy Clustering Algorithm for Images) using the texture features obtained from the normalized entropy function to detect spots in the skin lesions[3]. Nugroho *et al*. have classified melanin component of the skin to eumelanin and pheomelanin and proposed an image analysis method to quantify them[4]. These papers described pigmentation detection methods with stable lighting environment and complex detection algorithms.

To measure the skin condition accurately, it is necessary to get precise color data by cameras. But it is very difficult to get the correct skin color in the home because the skin color taken by the smartphones is different from the original skin by the influence of illumination. Fujitsu laboratory has proposed a color patch method for smartphones[5]. This method is implemented to get skin color data accurately in the home using smartphones. But this method cannot be applied

to measure skin color without correct lighting method. Various angles of illumination make different color when it applied to the reference color pattern. Therefore accurate camera illumination method must be considered.

Skin image-based measurement devices are currently used for hospital or beauty shops. It is so expensive to use them as a personal device in the home. These skin devices are implemented by a variety of imaging techniques such as a UV lighting and general illumination devices based on the condition of the skin.

Skin care products can be classified into stand-alone products and secondary products. The secondary products can be used with beauty devices or other health smart devices. These products are too expensive to use it as a personal device. This paper presents a design of low cost and easy-to-use device for a skin care system.

# 3 Skin care system design

## 3.1 Design requirements

The following functions are required to implement the skin care system on the smartphone.

① An external hardware device is required and it must have LED lighting, contact-type moisture sensors and communication capability to smartphones.
② Software programs are required to identify skin conditions such as pigmentation, wrinkles, pores, oily skin and so on.
③ An optical magnifying glass should be used to take skin images with lens zooming function.
④ It is required to measure skin types and quantify the measurement for dry, neural and oil level and so on.
⑤ It is required to save skin measurement data on the smartphones to use further detail analysis at a clinic. The application program provides users with easy user interface.

The hardware of the skin care system is designed with a microprocessor, sensors, white LEDs, UV LEDs, battery charger as shown in Figure 1. This device is connected to an Android smartphone via 4-wire earphone jack and communicates with smartphones. It receives commands and transmits the captured data of the skin. The transmission speed is 9600bps.

The wireless communication module is implemented using BLE(Bluetooth Low Energy), which is the lightweight version of Bluetooth 4.0. Since it operates very low power , it is applied to products for health care, fitness, beacons, securities and so on. Rechargeable battery is implemented to use the device as a portable skin diagnosis. The battery is 3.7v 300mAh lithium polymer battery with a circuit protection component to supply the stable power of the internal sensors and devices.

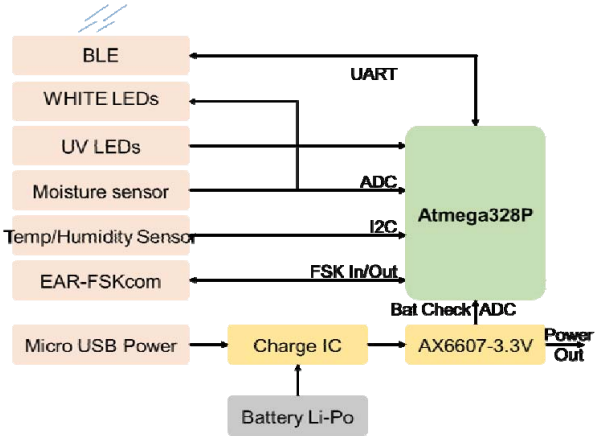



Figure 1. Block diagram of skin diagnosis device

## 3.2 Hardware design.

### 3.2.1 LED lighting

To measure skin condition correctly, two lighting systems are implemented. One is white light and the other is UV(ultraviolet) light. The UV LEDs are used to measure the oil content of the skin. CMOS sensors of the camera can determine oil detection by the amount of reflected waves. The white LEDs help to measure pigmentation, wrinkles, pores and so on. They emit white wavelength similar to that of natural light to the skin and are used to analyze and enlarge images of the skin. LEDs have been uniformly placed in a circle. Six white LEDs and six UV LEDs are placed one after another.

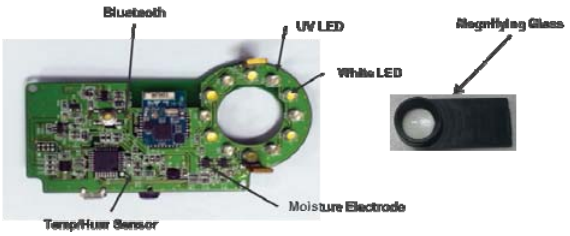



Figure 2. Designed PCB board

### 3.2.2 Sensors

The moisture of the skin is sampled by measuring current of the resistor. The moisture value is quantified to separate skin types such as dry, neutral and oily. The measured value is analog value and is converted to digital value by the processor. Temperature and humidity sensors are used as a secondary device to compensate the skin moisture by measuring the external temperature and humidity.


Figure 3. The skin diagnosis device mounted on a smartphone

#### 3.2.3 Processor

An ATmega328, a low power 8-bit RISC processor, is used to control the skin care system and to communicate with smartphones. Its internal flash memory is used to store the control and diagnosis program. The processor provides low power operation, AD converters and other interfaces in a single chip. Figure 2 shows PCB and Figure 3 shows the skin care hardware mounted on a smartphone.

### 3.3 Software design

#### 3.3.1 Firmware program

Firmware program for the ATmega processor initializes the system clock, LEDs, switches and so on. It initializes the BLE communication and the serial UART port. Firmware functional structure is divided into three parts, which are communication initialization, the main program and the sensor setup.

#### 3.3.2 Android program

The structure of the skin care Android application is programmed using C and libraries as shown in Figure 5. It has activities, database, BLE module and drivers. Algorithms for skin diagnosis have been implemented in C language. It is implemented as a library through JNI (Java Native Interface) included in the Android app. JNI can take advantage of the unique features of the device written in Java by API (Application Program Interface). Figure 6 shows the implemented skin care application on an Android smartphone and a captured image of the skin.

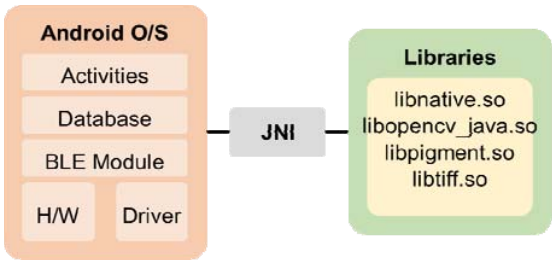



Figure 5. Android Application layer

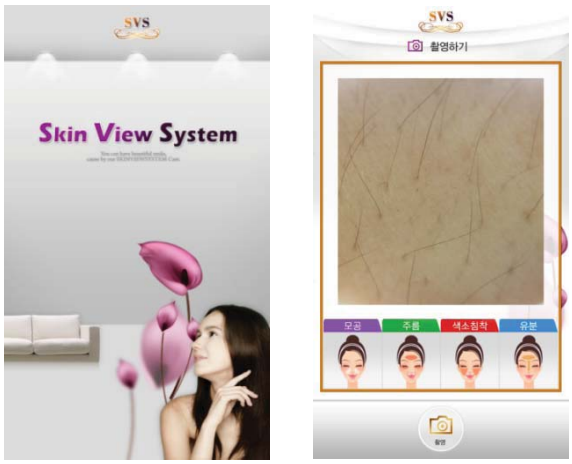



Figure 6. Android skin care system

#### 3.3.3 Skin diagnosis algorithm

Skin diagnosis algorithms can be divided into two parts. One is to measure skin condition such as pores, wrinkles, and moisture condition, and the other is to measure the skin pigmentation. These algorithms are shown as in Figure 7. Skin condition program uses the YCbCr color system to measure skin condition. It analyzes images by segmentation method and makes decision based upon the threshold value. After RGB images are converted to YcbCr images, Cb images are used to measure skin condition. The measurement process of oil level is very similar to skin condition but UV LEDs are used to measure oil level on the skin.

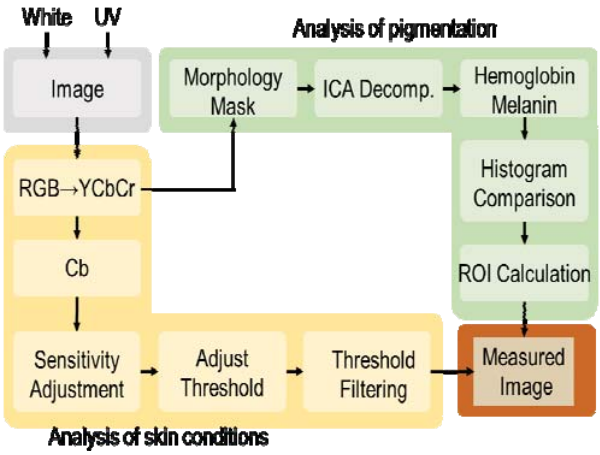



Figure 7. Image processing diagram

Expectation Maximization algorithm for Gaussian Mixture Model is applied to measure pigmentation by clustering skin color[6]. Cb and Cr components in the YCbCr color system are used to extract a skin area. After extracting skin area, ICA(independent component analysis) is applied to detect the hemoglobin and the melanin. Images are divided into two images, which are Cb image and Cr image and these images are used to calculate melanin level and hemoglobin level.

# 4 Measurement and analysis

In this section, the test results are analyzed to compare the skin care system with human diagnosis. Oily skin, moisture, and pigmentation measurements are discussed as below.

## 4.1 Oily skin measurement and accuracy

The processing example of oily skin is shown in Figure 8. The measurement process is shown in Figure 7. The accuracy of test results shows an average accuracy of 98.53 percent.

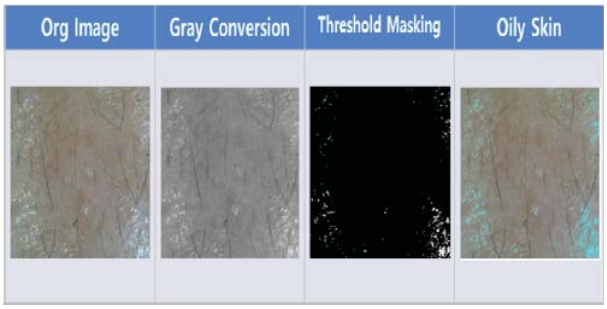



Figure 8. Processing example of oily skin

## 4.2 Moisture detection accuracy test

To measure the skin moisture, the skin care device uses a contact-type moisture sensor. It measures the fine current flowing to the skin surface by using resistors. To increase the accuracy, ten measurements are sampled to compare the moisture value for each resistance. The resistors are $10M\Omega$, $3M\Omega$, $1\ M\Omega$, $620K\Omega$, and $330K\Omega$. The lower resistor values show the higher accuracy as shown in Figure 9. Average accuracy for $330K\Omega$ is 98.54 percent and average accuracy for $10M\Omega$ is 97.2 percent.

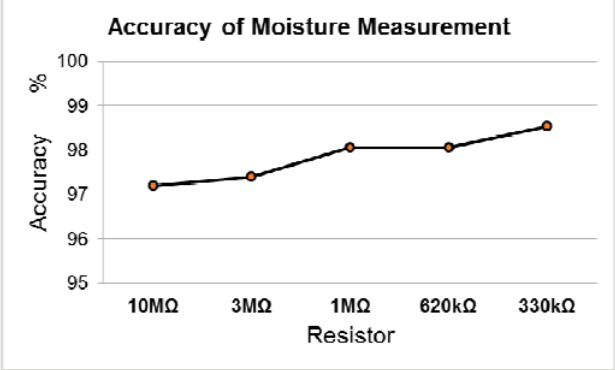



Figure 9. Accuracy by resistors

## 4.3 Pigmentation measurement and accuracy

Pigmentation measurement examples are shown in Figure 10. The pigmentation measuring process is shown in Figure 7. The accuracy of test results shows an average accuracy of 96.3 percent.

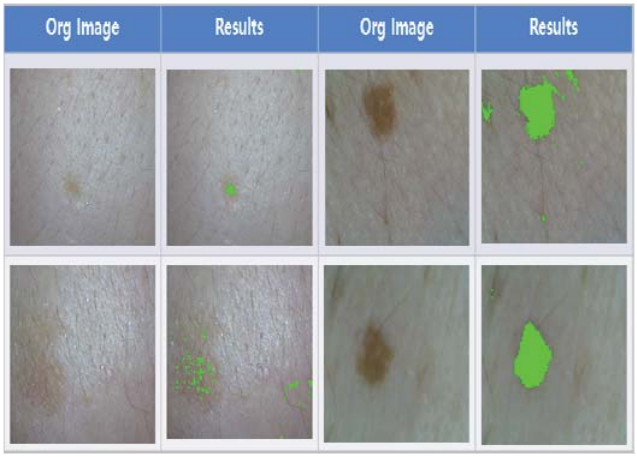



Figure 10. Pigmentation test examples

## 4.4 Processing time

Processing time for skin conditions is less than one second. The average time for oil detection is about 62.38*msec*. But the average processing time for the pigmentation program is about 3094.22*msec*. It takes relatively long time because it requires time for preprocessing, ICA processing, and image splitting and analyzing time to find color areas from images.

# 5 Conclusion

By attaching a skin care device, smartphones are used for a personal beauty care system to use personally in the

home. Pigmentation, wrinkles, pores can be measured based on the skin images. This paper has described the implementation of the skin care system. The detail hardware and software design has been discussed.

A camera on a smartphone captures skin images and sensors collect data for moisture, temperature and humidity by contacting directly to the skin. Smartphone application program processes the camera images and data from sensors and analyzes the skin status. The accuracy of the system is 96 percent or higher when it is compared with that of human examinations. The processing time is less than 1seconds for skin conditions such as wrinkles, pores, and oil content and so on. But the processing time of pigmentation takes about 3 seconds because of complex analyzing procedure.

Please address any questions related to this paper to Byoungchul Ahn by Email (b.ahn@yu.ac.kr).

## 6 References

[1] "Beauty Organization: A Global Strategic Business Report", Companies & Markets, 2013.

[2] "The US prestige beauty market through a new direct service trends", NPD Group, 2013.

[3] V.K Madasu and B.C. Lovell, "Blotch Detection in Pigmented Skin Kesions using Fuzzy Co-Clustering and Texture Segmentation," Digital Image Computing: Techniques and Applications, pp. 25-31, 2009.

[4] H. Nugroho, A.F.M. Hani, R. Jolivot, and F.Marzani, "Melanin Type and Concentration Determination using Inverse Model," National Postgraduate Conference (NPC), pp. 1-7. 2011.

[5] http://pr.fujitsu.com/jp/news/2012/05/7.html?nw=pr, "Skin measurement technology for smartphones. " Fujitsu

[6] Y. Ryu, S.H. Lee, S.G. Kwon, G. R. Kwon, "Skin pigmentation detection using projection transform block parameters," Journal of Korea Multimedia Society Vol. 9 No. 16, pp. 1044-1056, 2013.

# Early Output Hybrid Input Encoded Asynchronous Full Adder and Relative-Timed Ripple Carry Adder

P. Balasubramanian
School of Computer Science and Engineering
Nanyang Technological University
Singapore 639798
balasubramanian@ntu.edu.sg

K. Prasad
Department of Electrical and Electronic Engineering
Auckland University of Technology
Auckland 1142, New Zealand
krishnamachar.prasad@aut.ac.nz

***Abstract*—This paper presents a new early output hybrid input encoded asynchronous full adder designed using dual-rail and 1-of-4 delay-insensitive data codes. The proposed full adder when cascaded to form a ripple carry adder (RCA) necessitates the use of a small relative-timing assumption with respect to the internal carries, which is independent of the RCA size. The forward latency of the proposed hybrid input encoded full adder based RCA is data-dependent while its reverse latency is the least equaling the propagation delay of just one full adder. Compared to the best of the existing hybrid input encoded full adders based 32-bit RCAs, the proposed early output hybrid input encoded full adder based 32-bit RCA enables respective reductions in forward latency and area by 7.9% and 5.6% whilst dissipating the same average power; in terms of the theoretically computed cycle time, the latter reports a 10.9% reduction compared to the former.**

***Keywords*—Asynchronous design; Relative-timing; Indication; Ripple Carry Adder (RCA); CMOS; Standard cells**

## I. INTRODUCTION

Asynchronous circuit design using delay-insensitive data codes and a 4-phase return-to-zero (RTZ) handshake protocol is acclaimed to be a strong contender and/or a necessary supplement to mainstream synchronous circuit design by the International Technology Roadmap for Semiconductors (ITRS) design report [1]. Design for variability has been labeled as an important design challenge in the nanoscale electronics regime by the ITRS design report and in this backdrop, asynchronous circuit design based on delay-insensitive codes is attractive due to its inherent robustness to voltage, temperature and parameter variations [2] [3].

This paper presents the novel design of an early output hybrid input encoded asynchronous full adder which when cascaded to form a RCA results in less forward latency and cycle time and occupies less area compared to its existing counterparts whilst dissipating similar power. We shall first discuss some preliminaries before presenting the proposed asynchronous full adder. The dual-rail or 1-of-2 code is the simplest member of the generic family of delay-insensitive data codes [4]. In a dual-rail code, a valid data on a data wire W is represented using 2 data wires W1 and W0 as: W = 1 is represented by W1 = 1 and W0 = 0, and W = 0 is represented by W1 = 0 and W0 = 1; these two represent valid data. W1 = W0 = 0 is called the spacer, and W1 = W0 = 1 is invalid. The 1-of-4 code is used to encode two data wires (X and Y) using 4 data wires F0, F1, F2 and F3 as follows: X = Y = 0 is specified by F0 = 1; X = 0, Y = 1 is specified by F1 = 1; X = 1, Y = 0 is specified by F2 = 1 and X = Y = 1 is specified by F3 = 1. Only one of F0, F1, F2, F3 is asserted as 1 during the valid data phase. The spacer is represented by F0 to F3 all being 0s, and F0 to F3 cannot be all 1s simultaneously as it is invalid.

Strong-indication asynchronous circuits wait to receive all the input data before commencing data processing to produce the output data [5] [6]. Weak-indication asynchronous circuits tend to produce some output data after receiving even a subset of the input data but only after receiving all the input data, all the output data are produced [5] [7]. Early output asynchronous circuits could produce all the output data after receiving just a subset of the input data [8]. Early output asynchronous circuits can be further classified as early set or early reset type. If all the outputs of an early output asynchronous circuit acquire valid data after the application of just a subset of the valid inputs, it is said to be of early set type. On the other hand, if all the outputs of an early output asynchronous circuit assume the spacer state after the application of just a subset of the spacer inputs it is said to be of early reset type. The RTZ handshake protocol implies the RTZ of all the data wires (i.e. the assumption of the spacer state) after every application of valid input data [2].

## II. PROPOSED FULL ADDER – DESIGN AND OPERATION

Let (A0, A1), (B0, B1) and (CIN0, CIN1) denote the dual-rail full adder inputs, and (SUM0, SUM1), (COUT0, COUT1) denote the dual-rail full adder outputs. Hybrid input encoding implies the use of at least two delay-insensitive data encoding schemes, here, dual-rail and 1-of-4 codes for data encoding. The dual-rail augend and addend inputs of the full adder viz. (A0, A1) and (B0, B1) are 1-of-4 encoded, while the carry input, carry output and sum output are dual-rail encoded.

The equations governing the hybrid input encoded full adder are given by (1) to (4). Equations (1) to (4) are in disjoint sum of products/sum of disjoint products form [9] – [11], where the logical conjunction of any two product terms results in null (i.e. binary 0). Moreover, (1) to (4) satisfy the monotonic cover constraint thereby only one product term in a logical expression is activated at a time when valid input data is applied in the valid data phase.

$$SUM1 = E1CIN0 + E2CIN0 + E0CIN1 + E3CIN1 \quad (1)$$

$$SUM0 = E1CIN1 + E2CIN1 + E0CIN0 + E3CIN0 \quad (2)$$

$$COUT1 = E1CIN1 + E2CIN1 + E3 \quad (3)$$

$$COUT0 = E1CIN0 + E2CIN0 + E0 \quad (4)$$

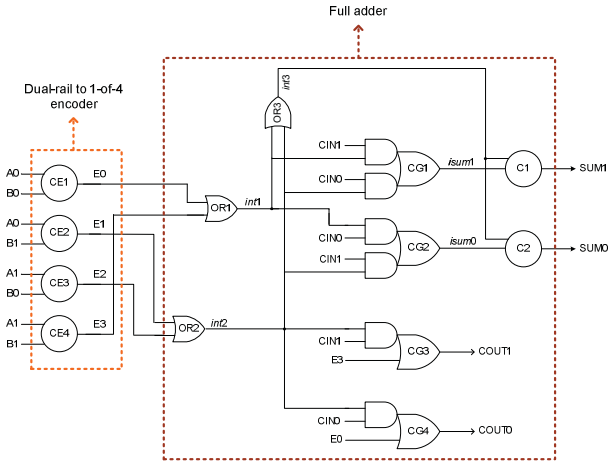



Fig. 1 Proposed hybrid input encoded early output asynchronous full adder

Fig. 1, which shows the proposed hybrid input encoded early output asynchronous full adder is the logic optimized synthesis of (1) to (4) by using simple and complex logic gates of a 32/28nm digital cell library [12]. The dual-rail to 1-of-4 encoder circuit is shown within the orange box in dotted lines, and the proposed full adder is shown within the brown box in dotted lines in Fig. 1. Logic redundancy [13] is implicit in the proposed full adder. In Fig. 1, CE1 to CE4 and C1, C2 represent 2-input C-elements. The C-element outputs 1 (0) when all its inputs are 1s (0s), and it maintains its existing steady-state otherwise. The 2-input C-element is realized using the AO222 complex gate with feedback. Gates CG1, CG2 are AO22 complex gates, and gates CG3, CG4 are AO21 complex gates. OR1, OR2 and OR3 are the simple gates. *int*1, *int*2, *int*3, *isum*1 and *isum*0 are the internal outputs. The internal outputs *isum*1 and *isum*0 are logically equivalent to the primary outputs SUM1 and SUM0 respectively. The operation of the proposed asynchronous full adder is described as follows by discussing carry-propagate, generate and kill modes with respect to Fig. 1.

### A. *Carry-propagate mode*

The carry-propagate mode is specified by A0 = B1 = 1, i.e. E1 = 1 or A1 = B0 = 1, i.e. E2 = 1. If E1 or E2 is 1 during the valid data phase, OR2 will output 1 on *int*2, followed by an output of 1 on *int*3. Depending on whether CIN0 or CIN1 is 1, CG1 or CG2 is activated and an output of 1 is produced on *isum*1 or *isum*0 respectively. Subsequent to the production of 1 on *isum*1 or *isum*0, since *int*3 is also 1, the primary output SUM1 or SUM0 then evaluates to 1. Also, since *int*2 is 1, depending on whether CIN0 or CIN1 is 1, COUT0 or COUT1 evaluates to 1. In the subsequent RTZ phase, if E1 or E2, whichever was 1 earlier returns to 0 after the RTZ of A0 and B1 or A1 and B0, *int*2 would RTZ, followed by *int*3 returning to 0. Now, regardless of whether CIN0 or CIN1 whichever was 1 earlier returning to 0, *isum*1 or *isum*0, whichever was 1 earlier returns to 0. Since *int*3 and *isum*1 or *isum*0 have returned to 0, the primary output SUM1 or SUM0 whichever was 1 earlier returns to 0. Further, after *int*2 has returned to 0, COUT1 or COUT0, whichever was 1 earlier, returns to 0. Hence, irrespective of the RTZ of the carry input (CIN0/CIN1), both the sum and carry outputs of the proposed full adder could RTZ thus demonstrating its early reset nature.

### B. *Carry-generate mode*

The carry-generate mode is specified by A1 = B1 = 1, i.e. E3 = 1. If E3 is 1 during the valid data phase, OR1 will output 1 on *int*1, followed by *int*3 outputting 1. Since *int*1 is 1 now, depending on whether CIN1 or CIN0 is asserted as 1, *isum*1 or *isum*0 is asserted as 1, followed by SUM1 or SUM0 being asserted as 1. Further, since E3 is 1, CG3 is activated and COUT1 is asserted as 1. In the following RTZ phase, once E3 returns to 0 after the RTZ of A1 and B1, *int*1 and subsequently *int*3 would also RTZ. Moreover, after *int*1 returns to 0, *isum*1 or *isum*0, whichever was 1 earlier would RTZ. This would be followed by SUM1 or SUM0 whichever was 1 earlier returning to 0. Also, after E3 returns to 0, COUT1 returns to 0. Hence, we find that the sum and carry outputs of the proposed full adder could RTZ regardless of the RTZ of the carry input thus demonstrating its early output viz. early reset nature.

### C. *Carry-kill mode*

The carry-kill mode is specified by A0 = B0 = 1, i.e. E0 = 1. If E0 is 1 during the valid data phase, OR1 will output 1 on *int*1, followed by an output of 1 on *int*3. After *int*1 becomes 1, depending on whether CIN1 or CIN0 is asserted as 1, *isum*1 or *isum*0 is asserted as 1, followed by SUM1 or SUM0 being asserted as 1. Further, since E0 is 1, CG4 is activated and COUT0 is asserted as 1. In the following RTZ phase, once E0 returns to 0 after the RTZ of A0 and B0, *int*1 and subsequently *int*3 also RTZ. Moreover, after *int*1 returns to 0, *isum*1 or *isum*0, whichever was 1 earlier would RTZ. This would be followed by SUM1 or SUM0 whichever was 1 earlier returning to 0. Also, after E0 returns to 0, COUT0 returns to 0. Thus, the sum and carry outputs of the proposed full adder could RTZ regardless of the RTZ of the carry input once again demonstrating its early output, i.e. early reset nature.

## III. Relative-Timed RCA

An important issue would arise when the proposed early output full adder is duplicated and cascaded to form an RCA, an example of which is shown in Fig. 2. Fig. 2 shows a 2-bit asynchronous RCA formed by cascading two stages of the proposed full adder. The dual-rail to 1-of-4 encoder circuits are not shown in Fig. 2 for the sake of simplicity and discussion. E0 to E3 are the 1-of-4 encoded data inputs, CIN01 and CIN00 is the dual-rail encoded carry input, and SUM01, SUM00 and COUT01 and COUT00 are the dual-rail encoded sum and carry outputs of the least significant full adder. On the other hand, E4 to E7 are the 1-of-4 encoded data inputs, COUT01 and COUT00 is the dual-rail encoded carry input, and SUM11, SUM10 and COUT11 and COUT10 are the dual-rail encoded sum and carry outputs of the most significant full adder.

In Fig. 2a, the red lines indicate a sample application of valid data inputs and the production of the corresponding valid data outputs during the valid data phase. Note that the carry-propagate mode is active in both the full adders since E1 and E5 are 1. In the least significant full adder, CIN00 is 1 and hence COUT00 becomes 1. In the most significant full adder, since COUT00 is 1, therefore COUT10 becomes 1. In Fig. 2b, the blue lines indicate an example partial RTZ of a subset of primary inputs of the 2-bit RCA and the following RTZ of all the primary outputs. It is assumed in Fig. 2b that E1 and E5 alone have returned to 0, but not the carry input CIN00 or the internal carry COUT00. It is seen that after E1 returns to 0 in the least significant full adder, OR2 outputs 0 and OR3 also outputs 0. CG1 outputs 0 and hence SUM01 returns to 0. After E5 returns to 0 in the most significant full adder, OR5 outputs 0 and OR6 also outputs 0. Since CG5 also outputs 0, SUM11 returns to 0. It is shown that CG8 outputs 0, i.e. COUT10 returns to 0. It is observed that with E1 and E5 alone returning to 0, and regardless of the RTZ of the primary carry input (CIN00) and the internal carry (COUT00), the primary sum and carry outputs of the 2-bit RCA RTZ reflecting early reset.

Given this, the late RTZ of CIN00 would not give rise to a wire orphan (i.e. unacknowledged transition on a wire) [8] [14] since the completion detector preceding the 2-bit RCA would indicate the RTZ of the primary carry input. The completion detector [2] is made up of an array of OR gates with a OR gate dedicated to combine the corresponding rails of an encoded input and the outputs of the array of such OR gates are combined using a C-element tree. The non-RTZ of the internal carry COUT00 might give rise to the problem of gate orphan [8] [14], where the gate orphan implies an unacknowledged transition on a gate output node. To overcome the likelihood of any gate orphan, a relative-timing assumption [15] is made in the region highlighted in Fig. 2b that the internal carry returns to 0 before the corresponding sum output returns to 0, i.e. the relative-timing assumption is that COUT00 returns to 0 before SUM11 returns to 0. This relative-timing assumption concerns maximum of only 2 full adder stages in any *n*-bit asynchronous RCA. So the 2-bit RCA in Fig. 2 is said to be relative-timed. However, despite the relative-timing assumption made, it is to be noted that the successive transitions within the circuit are all monotonic [16], i.e. there would be a wave of monotonically increasing (i.e. rising) transitions within the RCA during the valid data phase followed by a opposite wave of monotonically decreasing (i.e. RTZ) transitions within the RCA during the RTZ phase. Thus, imposition of the relative-timing assumption does not affect the input-output relation within the RCA.

To theoretically estimate the magnitude of relative-timing assumption to be made, we refer to the cell library information given in [12]. Note that we consider only the minimum size gates corresponding to [12] for this discussion. Referring to Fig. 2b, the critical path traversed for the direct RTZ of the sum output of a full adder consists of a 2-input OR gate viz. OR4 or OR5, an AO22 complex gate viz. CG5 or CG6, and a 2-input C-element viz. C7 or C8. The propagation delay associated with this critical path based on [12] is computed as 0.238ns. On the other hand, the critical path traversed for the indirect RTZ of the sum output of a full adder based on the carry input supplied from the preceding full adder consists of a 2-input OR

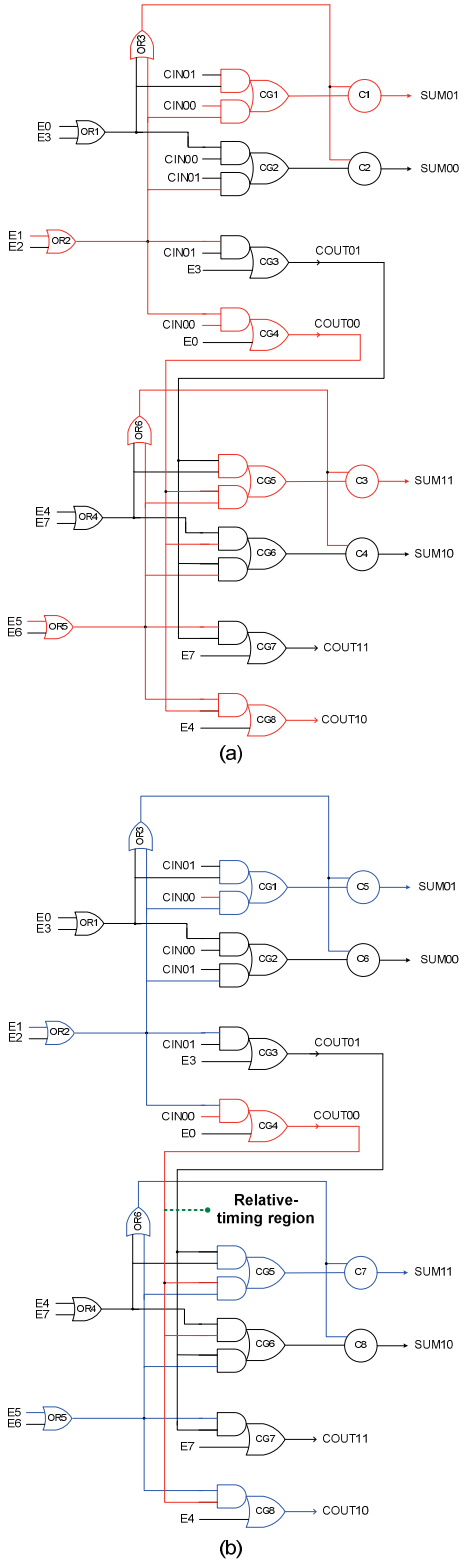



Fig. 2 A 2-bit relative-timed RCA constructed using the proposed early output hybrid input encoded asynchronous full adder

gate viz. OR2, an AO21 complex gate viz. CG3 or CG4, an AO22 complex gate viz. CG5 or CG6, and a 2-input C-element viz. C7 or C8. Hence the propagation delay encountered for the indirect RTZ of the sum output of a full adder stage is estimated to be 0.301ns. Therefore, there is a small negative timing slack of 0.063ns, which is the magnitude of relative-timing assumption required for any *n*-bit asynchronous RCA constructed using the proposed full adder. The negative timing slack may however be reduced by utilizing larger size i.e. high-speed gate(s) selectively for synthesizing the carry output logic of the proposed full adder.

## IV. Simulation Results and Conclusion

A number of 32-bit asynchronous RCAs based on hybrid input encoded full adders corresponding to [17] – [20] and the proposed full adder was constructed using the cell library elements of a 32/28nm CMOS process [12]. Safe quasi-delay-insensitive logic decomposition [21] wherever necessary was performed to ensure gate orphan freedom. More than 1000 random input vectors were supplied to the asynchronous RCAs at time intervals of 20ns through test benches. The .vcd files generated through the simulations were subsequently used to estimate the average power dissipation. The area and forward latency (i.e. critical path delay) of the asynchronous RCAs were also estimated using Synopsys tools. To calculate the cycle times of various asynchronous RCAs, their respective forward latencies were averaged and are multiplied by the corresponding time complexities of forward and reverse latency metrics given in [22]. In a similar manner, the cycle times of different asynchronous RCAs commensurate with their actual carry propagation length(s) can be computed.

TABLE 1. (FORWARD) LATENCY, CYCLE TIME, AVERAGE POWER, AND AREA PARAMETERS OF DIFFERENT HYBRID INPUT ENCODED 32-BIT ASYNCHRONOUS RCAS. THE RCA TYPE IS MENTIONED WITHIN BRACKETS IN THE 1ST COLUMN

| RCA and its type | Latency (ns) | Cycle time (ns) | Area ($\mu m^2$) | Power ($\mu W$) |
|---|---|---|---|---|
| Reference [17] (Strong-indication) | 9.24 | 18.48 | 2504.60 | 2179 |
| Reference [17] (Weak-indication) | 8.23 | 16.46 | 2423.27 | 2175 |
| Reference [18] (Strong-indication) | 9.22 | 18.44 | 2293.14 | 2171 |
| Reference [19] (Weak-indication) | 7.21 | 14.42 | 2016.63 | 2170 |
| Reference [20] – Non-redundant logic (Weak-indication) | 7.06 | 7.50 | 2016.63 | 2170 |
| Reference [13] – Redundant logic (Weak-indication) | 3.28 | 3.49 | 2049.16 | 2170 |
| This work (Relative-timed) | 3.02 | 3.11 | 1935.30 | 2173 |

From Table 1, it is clear that the latency, cycle time and area of the proposed full adder based asynchronous RCA are the least in comparison with the optimized latency, cycle time and area metrics of an RCA constructed using the full adder of [13] which explicitly features redundant logic – thanks due to relative-timing. With respect to average power, almost all the RCAs dissipate more or less an equal value, and this is because the full adders of all the RCAs satisfy the monotonic cover constraint [2], whereby specific signal path(s) are activated between the primary inputs and primary outputs for each input vector applied. Hence in comparison with its best competitor, the proposed full adder based asynchronous RCA enables respective reductions in forward latency, cycle time and area by 7.9%, 10.9% and 5.6%.

## References


[1] ITRS Design report. Available: http://www.itrs2.net

[2] J. Sparsø, S.B. Furber (Eds.), *Principles of Asynchronous Circuit Design: A Systems Perspective*, Kluwer Academic Publishers, 2001.

[3] P. Balasubramanian, *Self-Timed Logic and the Design of Self-Timed Adders*, PhD thesis, The University of Manchester, 2010.

[4] T. Verhoeff, "Delay-insensitive codes – an overview," *Distributed Computing*, vol. 3, no. 1, pp. 1-8, March 1988.

[5] C.L. Seitz, "System Timing" in *Introduction to VLSI Systems*, C.A. Mead and L.A. Conway (Eds.), Addison-Wesley Publishing, 1979.

[6] P. Balasubramanian, D.A. Edwards, "Efficient realization of strongly indicating function blocks," *Proc. IEEE ISVLSI*, pp. 429-432, 2008.

[7] P. Balasubramanian, D.A. Edwards, "A new design technique for weakly indicating function blocks," *Proc. IEEE DDECS*, pp. 116-121, 2008.

[8] P. Balasubramanian, "Comments on "Dual-rail asynchronous logic multi-level implementation," *Integration, the VLSI Journal*, vol. 52, no. 1, pp. 34-40, January 2016.

[9] P. Balasubramanian, D.A. Edwards, "Self-timed realization of combinational logic," *Proc. 19th IWLS*, pp. 55-62, 2010.

[10] P. Balasubramanian, R. Arisaka, H.R. Arabnia, "RB_DSOP: a rule based disjoint sum of products synthesis method," *Proc. 12th CDES*, pp. 39-43, 2012.

[11] P. Balasubramanian, N.E. Mastorakis, "A set theory based method to derive network reliability expressions of complex system topologies," *Proc. ACC*, pp. 108-114, 2010.

[12] Synopsys SAED_EDK32/28_CORE Databook, Revision 1.0.0, 2012.

[13] P. Balasubramanian, D.A. Edwards, W.B. Toms, "Redundant logic insertion and latency reduction in self-timed adders," *VLSI Design*, vol. 2012, Article ID 575389, pages 13, 2012.

[14] P. Balasubramanian, K. Prasad, N.E. Mastorakis, "Robust asynchronous implementation of Boolean functions on the basis of duality," *Proc. 14th WSEAS ICC*, pp. 37-43, 2010.

[15] K.S. Stevens *et al*., "Relative timing," *IEEE Trans. on VLSI Systems*, vol. 11, no. 1, pp. 129-140, February 2003.

[16] J. Cortadella *et al*., "Coping with the variability of combinational logic delays," *Proc. IEEE ICCD*, pp. 505-508, 2004.

[17] J. Sparsø, J. Staunstrup, "Delay-insensitive multi-ring structures," *Integration, the VLSI Journal*, vol. 15, no. 3, pp. 313-340, 1993.

[18] W.B. Toms, *Synthesis of Quasi-Delay-Insensitive Datapath Circuits*, PhD thesis, The University of Manchester, 2006.

[19] B. Folco *et al*., "Technology mapping for area optimized quasi delay insensitive circuits," *Proc. IFIP VLSI-SoC*, pp. 146-151, 2005.

[20] P. Balasubramanian, D.A. Edwards, C. Brej, "Self-timed full adder designs based on hybrid input encoding," *Proc. IEEE DDECS*, pp. 56-61, 2009.

[21] P. Balasubramanian, N.E. Mastorakis, "QDI decomposed DIMS method featuring homogeneous/heterogeneous data encoding," *Proc. ICDCC*, pp. 93-101, 2011.

[22] P. Balasubramanian, "A latency optimized biased implementation style weak-indication self-timed full adder," *Facta Universitatis, Series: Electronics and Energetics*, vol. 28, no. 4, pp. 657-671, December 2015.

# Emerging Approach to Infuse Catastrophe Model in Critical Real-Time Systems Management

**A.Christy Persya[1], T.R. Gopalakrishnan Nair[2]**
[1,2] Advanced Real Time Computing Group, RRGI Research Centre, VTU, Bangalore, India.
[2]Rector, RR Group of Institutions, Visiting Prof. NIAS, Bangalore, India.
[1]christypersya@gmail.com,[2]trgnair@gmail.com

**Abstract -** *In this paper, we present a new approach to detect and analyse the severity of catastrophe in real-time systems. In all critical real-time systems, there exists a definite probability that one or more of the subsystems can drift to conditions that spawn criticality. Three Mile Island, Chernobyl, and Fukushima Daiichi have demonstrated the occurrence of catastrophe due to various reasons and depicted the strange reaction produced by real-time systems which were not organized to handle and predict catastrophe chances. It is crucial to note that these systems were working satisfactorily in the normal mode before the decay time leading to the fully fledged accident. The excessive variations of values in the catastrophic domains disabled the system handlers that eventually resulted in a system breakdown. It is possible to enable a remedial capability in real-time systems to detect drift patterns of possible catastrophe indicators early and control the system to reduce the chance and scale of destruction. Here, we present a new methodology that incorporates a catastrophe model into real-time systems-so that the system has the capability to switch over to an efficient rescheduling mode when the possibility of catastrophe is detected at an early stage itself. Although the rescheduling strategy cannot completely avoid catastrophe but by detecting earlier, effects or severity can be reduced by giving warning alarms and enabling graceful degradation.*

**Keywords:** Real-time systems, cusp model, scenario based rescheduling, catastrophe indicator, criticality mode change.

## 1. Introduction

In any orderly designed real-time systems, such as a plant, there is an optimal scheduler capable of handling all the stipulated tasks with the schedulability criteria imposed on the system. This scheduler that is developed after extensive testing and compatibility analysis needs to accommodate a set of optional tasks under the slack period (domain) of the schedulability window. There can be several occasions for the plant when the normal scheduler gets challenged under demanding circumstances and meets the challenge by exercising the extra time and space given for execution. The concern is with this domain of functioning. While the scheduler is under the stretched performance of the real-time system, it may continue to perform satisfactorily. However, there could be hidden challenges up-way that may not be manifested by the task overload. There are several methods to deal with such situations. One of such conditions like overload is normally controlled by a strong admission policy of the scheduling algorithms. Overload conditions have been widely addressed in various research papers [1, 2, 3].

More recent work supporting scenario shift paradigm is criticality based system mode change [6]. When a system has more than one working mode, a protocol is required to handle the mode change smoothly [6, 7]. Here, three distinct types of modes are used to provide the functionality of the system. They are i) Normal Functional Mode ii) Exceptional Functional Mode iii) Degraded Functional Mode. In normal mode, the system application may move from one mode to other as part of its process like aircraft moving from cruise mode to landing mode. In exceptional mode, rare events cause the instability. But the response pattern is planned which can bring the system back to stability like when automated car detects an emergency scenario; it tightens the seat belt, and prepares to deploy the airbag. In third mode, i.e., degraded functional mode, the full set of error conditions may not be known prior which leads to uncertainty. The response pattern cannot be planned prior and online rescheduling is required like "fail code" mode in car.

The super scheduler model [4, 5] has two levels of the scheduler. Under normal mode, the primary scheduler schedules all the tasks, and in the event of an unexpected critical task, the outer scheduler has the ability to suspend the lower priority tasks and execute the most critical tasks. A priority alter protocol is required to alter the priorities of the low priority tasks and the newly entered critical tasks. Mixed Criticality (MC) systems use the terms criticality mode change [6]. Here, the mode change requires the continuous monitoring and rescheduling of tasks in the old mode and the new mode. The standard mixed criticality model [8, 9, 10, 11] addresses a path

from low criticality level (LO) to high criticality level (HI). The system is in LO criticality mode initially. According to the MC schedulability condition [21], if any LO criticality task executes for its Worst Case Execution Time (WCET), without completing the job, then the system moves to HI criticality mode. Here, the task executes for its HI criticality WCET by aborting the LO criticality jobs. An analysis on the state where the LO criticality tasks differed/discarded while the HI criticality jobs are executed. Related approaches include elastic task model [22], MC fluid task model [23], robust priority assignment approach [24], and bailout protocol [25]. The aim of these approaches is to reduce the number of LO criticality jobs to be discarded and restore LO criticality mode following an interval of HI criticality activity in exceptional functional mode.

Adaptive mode changes are planned, but usually they don't occur. There can be a situation wherein the mode change is not planned for and is unpredictable, such as in degraded functionality mode. In this paper, we present novel modelling approaches based on the well-known catastrophe theory that would enable efficient tracking and detection of an impending catastrophe in the above-mentioned situations.

The paper is structured as follows. Section 2 explains the geometry of catastrophe to be detected and analyzes its severity. Section 3 introduces the improvement that allows incorporating the catastrophe theory into the real-time systems. Section 4 concludes the paper with a future scope.

## 2. Geometry of catastrophe theory

In certain situations where stretched scheduling takes place, the bifurcation of states has a definite probability to occur. In the case of complex plants engaged in fairly unstable process management like nuclear power plant, aircraft and rocket control, war scenarios, complex process control systems, automated monitoring systems, space rovers, and fully automated vehicles, the detection of bifurcation offers a good opportunity to take proactive measures. Hence, methods of detecting such scenario shift emanating from probable bifurcations of states of the plant demand heavy attention in creating intelligent real-time systems (RTS). Focus on such scenarios requires the development of scenario shift and bifurcation analysis incorporated into real-time core management aspects like intelligent schedulers. Here, we discuss the requirements and conditions for implementing such intelligent approach in practical applications. They include identification of the state in the scenario with respect to the state capable of producing the bifurcation.

In order to consider the reality, we need to generate the scenario models and detect such drifts or bifurcation of states in plants. This information can be used to alter the real-time systems performance. For this purpose, we require the standard approaches of dealing catastrophe through modelling and analysis and implementing such practises into the real-time system design domains. These techniques involve adapting bifurcation geometries into scenario representing functions. The standard catastrophe geometries [12] is given in Table1.

Table 1. Geometries of Catastrophe

| TYPE | EQUATION |
|---|---|
| Fold Catastrophe | $V_a(x) = \frac{1}{3}x^3 + ax$ |
| Cusp Catastrophe | $V_{ab}(x) = \frac{1}{4}x^4 + \frac{1}{2}ax^2 + bx$ |
| Swallowtail Catastrophe | $V_{abc}(x) = \frac{1}{5}x^5 + \frac{a}{3}x^3 + \frac{b}{2}x^2 + cx$ |
| Butterfly Catastrophe | $V_{abcd}(x) = \frac{1}{6}x^6 + \frac{a}{4}x^4 + \frac{b}{3}x^3 + \frac{c}{2}x^2 + dx$ |

The goal of the catastrophe theory is to specify a set of common criteria for discontinuity hypothesis. It classifies the various ways in which a system can undergo sudden large changes in behaviour as one or more of the variables that control it are changed continuously. Catastrophe is defined as the sudden changes that are caused by the smooth alterations in the situation. E.g. melting of ice or boiling of water. In more general terms, a small increase in stress leads to a dramatic change in behaviour. This sudden change in the system described by Thom [13] may lead to an abrupt or major change in the stability of the system. Catastrophe theory explains the regularities in a mathematical structure which permit routine calculations of how systems behave, based on the Taylor series approximation. The catastrophe theory described by Poston and Steward mainly focus on the geometric and algebraic methods used to handle Taylor series properly [12]. They have described seven elementary catastrophes to address the sudden jump and discontinuity. Gilmore proposed the indicators of discontinuities as the phenomena of bimodality, inaccessibility, hysteresis, and divergence [26]. These catastrophe flags indicate the presence of catastrophe. There are two other variants in catastrophe handling. One of

these is the Gilmore's catastrophe detection. This detection involves the application of all catastrophe flags. A second variant is the catastrophe analysis consists of a mathematical analysis of the dynamic equations of a transition process. A catastrophe model is an intermediate between the detection and analysis. The more the catastrophe flags are detected, the more convincing is the claim of a catastrophe. Smooth changes in independent or control variables of a system may lead to abrupt or discontinuous changes in the system. Among the seven elementary catastrophe model, cusp model is the simplest model that is used in many applications. The cusp model gives rise to sudden discontinuities and is based on the non-linear deterministic dynamic system. It consists two control variables and one behavioural variable.

Most of the catastrophe associated with practical applications fall into elementary catastrophe regions, and many can be dealt with the geometrical structures presented above. The beautiful way of thinking about the geometry of the higher-dimensional cases is described in callahen[14].

To study the applicability of catastrophe theory in a plant, it is imperative to know its operational features. Here, we consider the generalized property of a typical plant that provides a catastrophe shape with folding functions prevailing in nature.

A cusp catastrophe folding function can be given as

$$V_{ab}(x) = \frac{1}{4}x^4 + \frac{1}{2}ax^2 + bx \qquad (1)$$

In such circumstances, the real-time scheduling module can be augmented by the analytical output of the catastrophe model to enhance the decision-making capability of the real-time scheduler. The arrival of unpredictable tasks, due to the variations in parameters- in some cases, can lead to a sequence of failures amounting to catastrophe, which in turn induces instability in the system. There have been a few reports that attempted to use the catastrophe theory to detect the stability of the system [15, 16, 17]. Here, we propose a general methodology for detecting such variations at an early stage that would be a critical functionality of next generation RTS which would be able to detect and analyze a catastrophe through causal variables.

In [15], catastrophe theory was used in conjunction with the dynamic workload allocation algorithms to observe their dynamic behaviour. Similarly in RTS, this methodology can enable detection of the threshold level up to which the system can be considered safe or to identify the instant when the system should be shut down in order to avoid high level damage. The instability of the system can be caused because of several unexpected behaviour changes in tasks that result in missing the deadlines of several critical tasks.

The balanced surface of $V_{ab}(x)$ is

$$x^3 + ax + b = 0 \qquad (2)$$

The singular point of $V_{ab}(x)$ is

$$3x^2 + a = 0 \qquad (3)$$

The first derivative of functions depicts the balanced surface of the curve which probably will not pass through the origin and the second derivative of the function shows the singular point through which a branch of line passes through the origin.

An optimal or feasible real-time schedule can be stressed or stretched by the drifting scenarios manifested through several parameters expressed in real-time systems performance. This drift can be detected and analyzed by the catastrophic model that can be concurrently executed with it. This will identify the property of variation of the parameters involved in the operational model.

## 3. Catastrophe theory in RTS

The scenario shift in the real-time system happens when there is a major change in the system variables and environment variables. The major change could be the result of the smooth transition. The catastrophe indicators analyze the critical points and prove the presence of catastrophe. Here, catastrophe is meant as the bifurcation of a smooth transition of control variables leading to the sudden change in behaviour. A smooth function will be guiding a normal tightly packed hybrid scheduler in any real-time system. The effect of second order derivative determines the critical point. When its derivative becomes zero, a smooth transition occurs. A system is considered to have more than one variable. A function governing the scenario shift will have a bounded limited dispersion. The control variables try to adjust to the changes. However, when changes become unbounded, the scheduler calls for scenario shift. The input variables shared by smooth transition and sudden jump catastrophe will be same. That is, the variation in control variables in bifurcation leads to the sudden jump catastrophe. All variables involved in normal scheduling will also be involved in catastrophe scenario. When bifurcation happens, the catastrophe cannot be avoided, which eventually leads to gradual degradation of

performance that are controlled by activities such as power shut down, etc.

In the general scenario, the real-time system will enable the normal mode in which the catastrophe modeler determines the paring of variables in a cusp model. Several cusp will be ensemble for catastrophe detection. Among these any of the $i^{th}$ cusp (where i=1,2,3,….,n) can initiate the catastrophe which can be realized by the catastrophe indicators. The cusp model is specified with two control variables and one behavior variable. When there are N total normal variables and m control variables in a system, m/2 cusp is required for catastrophe detection. In a domain, N-m scheduler will work perfectly.

The ensemble cusp for m variables is shown in Figure 1.

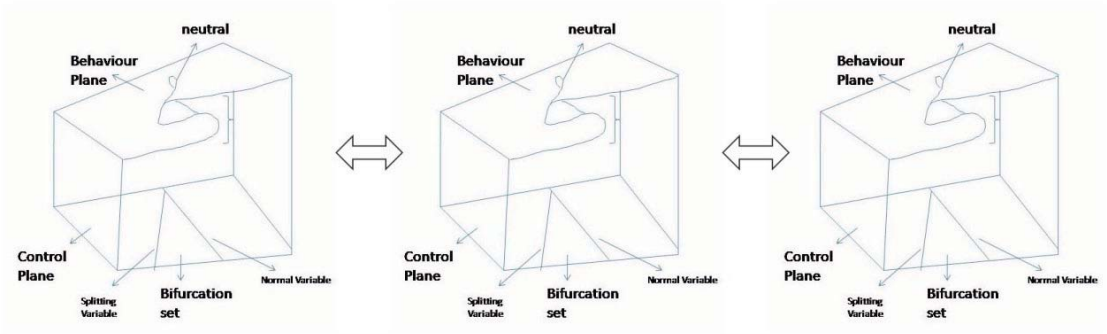



Figure 1. Ensemble of cusp

Here, each $i^{th}$ cusp in a m/2 cusp model has got an order and priority of criticality which depends on the built-up function. High rate point of inflexion and divergence will be very high. Initially divergence can be controlled and then the remaining part. When it is realized that the situation cannot be handled, then the $i^{th}$ cusp in m/2 cusp model with its priority ordering will be moving through highest priority first and lowest priority last. Moreover, to deal that the scheduler will have a pure preemptive priority-driven scheduling algorithm.

**Ensemble of functions making cusps**

$$f(x,a,b) = x^4 + bx^2 + cx \quad (4)$$

The equation of the balanced surface is

$$4x^3 + 2bx + c = 0 \quad (5)$$

The equation of the singular point set is

$$12x^2 \mid 2b = 0 \quad (6)$$

Here, according to the catastrophic principle, the difference set equation and the control variables affecting the catastrophe are described by means of cusp model.

The life of a plant is well explained in Figure 2. The real-time task and control domain explains the normal working plan of any real-time scheduler. According to acceptance test and schedulability test, all hard deadline tasks complete their execution. So the positive operation of real-time task index will be 100% in normal scenario. When there is a raise in smooth function observed in catastrophe domain, the hessian determinant tells the point of inflexion. Because of the unexpected scenario shift, few tasks may be added. So the negative operation of real-time task index deviates from 100% down to 10% . If the catastrophe raise is not handled using rescheduling strategies, it leads the system to failure.

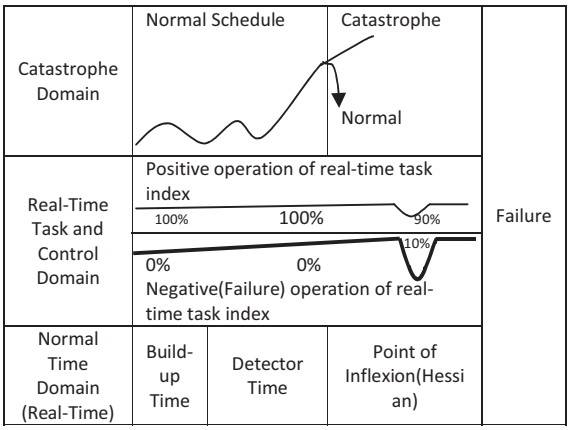



Figure 2. Life of the System

A critical real-time system consists of several subsystems and each subsystem behavior is considered as a smooth function. In any smooth function, there are chances for critical points (non degenerate) in which the derivative is zero. In

general there are only two critical points: minima and maxima. Morse Lemma explained it as nice critical points (more than one variable) and nasty critical points (one variable) [12]. Splitting lemma helps in reduction of number of variables. Hessian determines the local behavior of the function. So a critical point, in which the hessian matrix is non singular, it is said to be nondegenerate. The pseudo code for incorporating catastrophe theory in RTS is given below

```
Program cata-detec;

let     T1,T2,....Tn set of tasks;
        V_ab(x) the function of 2 control variables;
        Σ V_ab(x) ensemble of cusp with 2 control
variables;

begin
At regular intervals,
    for each flexible schedule do
        for each two control variables do
                Develop cusp model;
                Make ensemble of cusps;
                Analyse the behaviour based on
                    the control variables;
                Detect catastrophic drift through
                    models when the control
                    variables are in bifurcation
                    region;
                Call severity_check();

End cata-detec;
```

```
Function severity_check()

begin
        Identify critical points in the smooth
function;
        Develop the Hessian matrix;
        Find the Hessian determinant;
        if determinant = = 0 then
                do rescheduling plan;
                wait (time);
                if determinant becomes < or > 0
                        then
                        continue normal-mode;
                        else
                        shift to superscheduler
mode;
                endif
        endif

end severity-detec;
```

RTS are designed for certainty. There is no built-in approach till now that can detect combinatorial uncertainty, which may arise due to unknown under performance or failures. Catastrophe theory provides some capability to detect the drift. The cusp catastrophe model is chosen because it is simple and easy for explaining the nonlinear relationships. When there is overload, few optional tasks are identified and discarded. The arrival of unexpected task entry or changes due to variations in parameter or sudden shut down due to external environment leads the system to catastrophe mode.

Various aspects of the criticality mode are discussed in [6]. In certain instances, catastrophic shift is required to maintain the stability of the system. So, the system needs to identify the situation where the criticality mode change is required and when the catastrophe shift can happen. The mixed criticality mode change deals with a planned scenario whereas the superscheduler is expected to work with an unplanned situation. Since the catastrophe does not arise in a planned way, it is difficult to maintain the true stability of the system. Under such circumstances, the optimal solution is to manage the situation for the best benefits. There are three options: (i) partial rescheduling (ii) complete rescheduling (iii) graceful degradation [18]. Here, the catastrophe theory plays a major role in detecting the situation in which the superscheduler is brought into action.

The critical situation can be transient overload or overload leading to catastrophe. So the system can be alerted about the variation in parameters and possible rescheduling can be planed during the raise in function from the normal scenario. A limited level change can be tolerated expecting transient fault [19, 20]. When the threshold is reached, the rescheduling plan is executed immediately. The costs of computation involved in planning rescheduling are much less than the effects of catastrophe. In rescheduling, few lower priority tasks are discarded and various effective techniques can be used to design rescheduling strategies [15].

The rescheduling strategy cannot completely avoid catastrophe but by detecting earlier, effects or severity can be reduced by giving warning alarms and enabling graceful degradation.
For any real-time systems, the software component apart from scheduling must be capable of designing catastrophe modeling and catastrophe detection method and retrieval strategies to maintain the stability of the system. Cusp model is one of the approaches proposed here with two control variables or splitting variable and normal variable. The ensemble of cusp is required in a real-time

system such that any two variables can make a cusp.

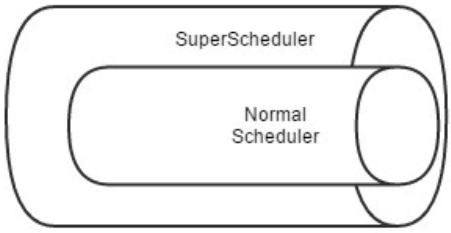



Figure 3. Normal Scheduler within super scheduler

The normal scheduler or the inner scheduler takes care of the planned events or tasks, and the stretched or the superscheduler works for the unplanned mode changes by rescheduling the tasks mainly using priority alter protocol [4, 5]. This rescheduling of tasks can be partial or complete based on the severity of detected catastrophe. Therefore, the unexpected arrival of a normal task may trigger another event and thus can create a chain of tasks that might lead to catastrophe. The smooth function having a minimum, maximum, or the point of inflexion is considered to be stable property [12]. The invariant raise, for example, caused by a task being executed for a longer period than expected could be a transient or permanent raise. Here, the super scheduler could plan for the feasible rescheduling. When the catastrophe indicators strongly depict the presence of unusual critical behaviour, then the rescheduling plan can be executed. Otherwise the system can continue in the normal mode. By doing this rescheduling plan at an earlier stage, in case of transient then the computation time of plan will be discarded. When compared to the disaster itself, this computation time can be considered as a preventive measure. The catastrophe failure could be detected as transient or permanent. Transient raise in smooth function will drop down and fall under the normal mode after few minutes. But the permanent failure keeps building the chain of failure to cause disaster. This is shown in Table 2.

Table 2. Rescheduling Category

| Failure leading category | Rescheduling Plan | Reschedule Execution |
|---|---|---|
| Transient | Done | No |
| Permanent | Done | Done |

## 4. Conclusion

In this paper, we have proposed a new improvement for real-time systems by choosing some of the features of the catastrophe theory to detect the unusual changes happening in the system and analyze the severity of the changes. The normal conditions in the system are dealt by the usual hybrid scheduler. When the unusual changes are detected by the catastrophic observer, the system can no more be handled in the same way when compared to overload situations. In such situations, the raise in function must be continuously monitored and rescheduling plans must be created. The cusp model has been used to detect the catastrophe trend in the function with two control variables and one behaviour variable. As a part of the future work we plan to make an ensemble of cusp in a system to detect the unusual changes and work out the reschedulablity analysis in more detail.

## References

[1.] Cheng, Zhuo, et al. "Greedy scheduling with feedback control for overloaded real-time systems." International Symposium on Integrated Network Management (IM), 2015 IFIP/IEEE, IEEE, 2015.

[2.] Springer, Thomas, Steffen Peter, and Tony Givargis. "Resource Synchronization in Hierarchically Scheduled Real-Time Systems using Preemptive Critical Sections." IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2014. IEEE, 2014.

[3.] Stankovic, John A., et al. Deadline scheduling for real-time systems: EDF and related algorithms. Vol. 460. Springer Science & Business Media, 2012.

[4.] T.R.Gopalakrishnan Nair, A. Christy Persya, "Critical Task Re-assignment Under Hybrid Scheduling Approach in Multiprocessor Real-time Systems", Parallel and Distributed Computing and Systems, ~PDCS 2011~, The 23rd IASTED International Conference on. pp. 130-137, December 14 – 16, 2011, Dallas, USA. DOI: 10.2316/P.2011.757-071

[5.] Persya, A. Christy, and TR Gopalakrishnan Nair. "Model based design of super schedulers managing catastrophic scenario in hard real time systems."Information Communication and Embedded Systems (ICICES), 2013 International Conference on. IEEE, 2013.

[6.] Burns, Alan. "System mode changes—general and criticality-based." Proc. 2nd Workshop on Mixed Criticality Systems (WMC), RTSS. 2014.

[7.] Ekberg, Pontus, and Wang Yi. "Schedulability analysis of a graph-based task model for mixed-criticality

systems." Real-Time Systems (2015): 1-37.

[8.] Baruah, Sanjoy K., Alan Burns, and Robert I. Davis. "Response-time analysis for mixed criticality systems." Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd. IEEE, 2011.

[9.] De Niz, Dionisio, Karthik Lakshmanan, and Ragunathan Rajkumar. "On the scheduling of mixed-criticality real-time task sets." Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE. IEEE, 2009.

[10.] Burns, Alan, and Robert Davis. "Mixed criticality systems-a review."Department of Computer Science, University of York, Tech. Rep (2013).

[11.] Li, Haohan, and Sanjoy Baruah. "An algorithm for scheduling certifiable mixed-criticality sporadic task systems." Real-Time Systems Symposium (RTSS), 2010 IEEE 31st. IEEE, 2010.

[12.] Poston, Tim, and Ian Stewart. Catastrophe theory and its applications. Courier Corporation, 2014.

[13.] Thom, René. "Structural stability and morphogenesis." (1989).

[14.] Callahan, J. "Geometry of E6 and anorexia." Preprint, Mathematics Institute, University of Warwick, UK (1977).

[15.] Schreiber, Fabio A., et al. "A study of the dynamic behaviour of some workload allocation algorithms by means of catastrophe theory." Journal of systems architecture 43.9 (1997): 605-624.

[16.] Van der Maas, Han L., and Peter C. Molenaar. "Stagewise cognitive development: an application of catastrophe theory." Psychological review 99.3 (1992): 395.

[17.] Yue, Yang, et al. "Network traffic anomaly detection method based on a feature of catastrophe theory." Chinese Physics Letters 27.6 (2010): 060501.

[18.] Gopalakrishnan Nair, T.R.; Christy Persya A., "Catastrophe Detection And Analysis for Recasting Real Time Scheduling Strategies For Effective Management," Advances in Engineering and Technology, Proceedings of the Fourth International Joint Conference on, Volume No 6, pp. 492-496, NCR ,India, Dec 13-14, 2013. SearchDL ID: 03.elsevierst.2013.6.103.

[19.] Baruah, S., A. Burns, and R. I. Davis. "An extended fixed priority scheme for mixed criticality systems." Proc. ReTiMiCS, RTCSA (2013): 18-24.

[20.] Acharya, Subrata, and Rabi N. Mahapatra. "A dynamic slack management technique for real-time distributed embedded systems." Computers, IEEE Transactions on 57.2 (2008): 215-230.

[21.] Baruah, Sunandan, et al. "Scheduling real-time mixed-criticality jobs."Computers, IEEE Transactions on 61.8 (2012): 1140-1152.

[22.] Su, Hang, and Dakai Zhu. "An elastic mixed-criticality task model and its scheduling algorithm." Proceedings of the Conference on Design, Automation and Test in Europe. EDA Consortium, 2013.

[23.] Baruah, Sanjoy, Arvind Eswaran, and Zhishan Guo. "MC-Fluid: simplified and optimally quantified." Real-Time Systems Symposium, 2015 IEEE. IEEE, 2015.

[24.] R.I. Davis and A. Burns. Robust priority assignment for fixed priority real-time systems. In Proc. of IEEE Real-Time Systems Symposium, pages 3–14, 2007.

[25.] Bate, Iain, Alan Burns, and Robert I. Davis. "A bailout protocol for mixed criticality systems." Real-Time Systems (ECRTS), 2015 27th Euromicro Conference on. IEEE, 2015.

[26.] Gilmore, Robert. "Catastrophe theory." Encyclopedia of applied physics 3 (1992): 85-115.

# An Efficient Traffic-Based Routing Algorithm for 3D Networks-on-Chip

Hsueh-Wen Tseng[1], Ruei-Yu Wu[2*], Wan-Chi Chang[1], Yi-Huo Lin[1], and Dyi-Rong Duh[3]
[1]Department of Computer Science and Engineering, National Chung-Hsing University, Taiwan
[2]Department of Management Information Systems, Hwa Hsia University of Technology, Taiwan
[3]Department of Computer Science and Engineering, Hwa Hsia University of Technology, Taiwan

## Abstract

Network-on-Chip (NoC) is proposed to solve the communication challenges in multiprocessor System-on-Chip architecture. 3D NoC uses through silicon vias to stack multiple 2D silicon layers and obtain better performance. However, 3D NoC has to design an efficiently adaptive routing algorithm to overcome huge traffic load problems. An adaptive routing algorithm is composed of routing function (RF) and selection function (SF). RF decides a set of deadlock-free candidate paths, and SF chooses a proper routing path from those candidates. In this paper, we propose a novel traffic-based routing algorithm, TBRA, to achieve better performance. TBRA combines some advantages from different routing algorithms and selects adaptive routing algorithms according to the traffic load. Furthermore, it also collects two hops data of networks to determine the better routing path. Simulation results show that the proposed scheme significantly improves the transmission latency, the throughput, and the energy consumption.

**Keywords:** Network-on-Chip, 3D IC, Congestion, Selection Function, Routing Algorithm.

## 1. Introduction

Nowadays, multiprocessor System-on-Chip (MP-SoC) integrates many functional blocks onto a single die [1] and the communication becomes bottleneck between two processor elements (PEs). Network-on-chip (NoC) is proposed to provide a scalable packet switched interconnection architecture for on-chip communication solutions. However, 2D chip architecture significantly increases wire delay and power consumption in the deep submicron regime [2], [3]. 3D NoC vertically stacks many 2D dies by using through-silicon-vias (TSVs) [4], [5] and has shorter global interconnection length, higher performance, and better scalability.

However, 3D NoC generates more traffic load than conventional planar ICs [6]. Higher traffic load easily results in traffic congestion. Conventional planar routing algorithms never suffer such huge traffic load and cannot handle the huge traffic load problem. Typical routing algorithms are classified as deterministic routing algorithms and adaptive routing algorithms based on the path selection process [7]. Deterministic routing algorithms (e.g., XY routing algorithm [8]) use a simple router architecture and predetermine a routing path with deadlock free. Although they obtain lower transmission latency in light traffic load, they cannot avoid the occurrence of hotspots and congestion paths as the packet injection rate (PIR) increases.

On the contrary, adaptive routing algorithms consider the traffic variations of NoC and dynamically calculate proper routing paths to avoid the hotspots and congestion areas [9]. The selection of routing path is determined from the current status of NoC (e.g., congestion paths, hotspot regions, and blocking time). Although adaptive routing algorithms pay slightly higher transmission latency than deterministic routing algorithms in low traffic load due to the hardware overhead of routing implementation, they work well in high traffic load.

Previous works for selection function (SF) only focused on a routing path decision based on the buffer information of adjacent routers or blocking time [10]. However, most of them only took into account how to avoid the adjacent congestion areas when they forwarded packets to destination nodes [11], [12]. These papers only used a partial network information to choose a proper routing path, so packets easily blocked during transmission.

In a general adaptive routing algorithm, a PE generates a packet and the packet is sent to a router for transmitting to a destination. The router uses RF to compute a number of routing paths to forward the packet, and then SF chooses an appropriate output channel (OC) according to network information (i.e., the buffer information of adjacent routers). Once RF selects incorrect routing paths to SF, SF cannot determine the suitable routing path. As a result, it results in traffic congestion problem.

In this paper, we propose a traffic-based routing algorithm (TBRA) to dynamically select suitable adaptive routing algorithms according to the network situations. TBRA consists of a feedback traffic-based routing function (FTBRF) and a traffic-based selection function (TBSF). TBSF uses network information of two hops to precisely determine the proper output port. When TBSF detects the bad situations about the input buffers of neighbors with two hops, TBSF informs FTBRF to dynamically switch routing algorithms depending

*The corresponding author. E-mail: d92007@csie.ntu.edu.tw

on the network situations.

This paper is organized as follows. In Section 2, we describe the related work. Section 3 describes the proposed traffic-based routing algorithm, TBRA. Then, we show our simulation results in Section 4. The conclusions are shown in Section 5.

## 2. Related work

In this section, we introduce the conventional adaptive routing algorithms and selection function schemes, respectively. The adaptive routing algorithms are classified as fully adaptive routing algorithms and partially adaptive routing algorithms based on the number of selected paths between the source node (S) and the destination node (D). A fully adaptive routing algorithm is presented that a packet is transmitted to use all possible paths from S to D. It has the highest adaptiveness but yields the risk of deadlock.

On the contrary, partially adaptive routing algorithms only use a subset of possible paths to route packets. They have lower adaptiveness and deadlock-free. The turn model [13] and the odd even model [9] are famous partially adaptive routing algorithms without virtual channels. Turn model algorithm limits the minimum number of turns to avoid deadlock, and OE routing is proven to have higher even adaptiveness than the turn model routing. However, OE results in imbalance traffic and then incurs congestion problems in different routing paths. Thus, it is an important issue to design a good selection function to assist adaptive routing algorithms in NoC.

Most of conventional selection functions referred to congestion information, such as Dynamic XY (DyXY) [12], Neighbors-on-Path (NoP) [11], and BARP [13]. They used congestion-aware algorithms to analysis the congestion status of the current router and adjacent routers. According to the obtained network information, each router determines a suitable OC to transmit a packet. DyAD adopted the information of buffer capacity to alleviate traffic congestion among neighboring routers [15]. Although DyAD dynamically switched deterministic and adaptive routing algorithms to alleviate traffic congestion, it only considered the adjacent network information and easily caused misjudgment problems. In literature [16], the authors both consider run time contention and bandwidth-aware selection function in NoC. In literature [17], they proposed a destination-based selection strategy (DBSS) to integrate local and global network information and avoid network congestion. However, these schemes required specific hardware design and only used in 2D NoC.

Briefly, most of previous selection schemes only considered the buffer information of adjacent routers while S sent packets to D. However, these schemes may determine an incorrect routing path and suffer traffic congestion during transmission as shown in Fig. 1.

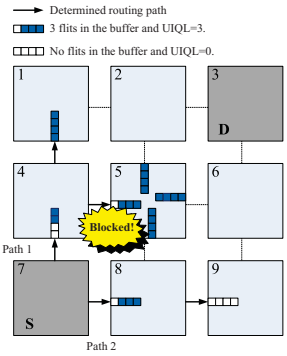



Fig. 1: Misjudgment problems of conventional selection function schemes.

Fig. 1 illustrates misjudgment problems in a mesh NoC. We assume the design of the router uses wormhole scheme, and the input buffer size is set to 4 flits without virtual channel (VC). When S wants to transmit packets, it considers the buffer information of the adjacent routers. There are two candidate paths: path 1 and path 2. Path 1 uses router 4 to forward packets, and path 2 uses router 8 to forward packets. The used input queue lengths (UIQLs) of router 4 and router 8 are 2 and 3. Hence, S selects path 1 to forward packets. There is a similar procedure in router 4. Router 4 finds the UIQLs of router 1 and router 5 are 4 and 3. Hence, router 4 forwards packets to router 5. However, router 5 suffers congestion, and packets are blocked in router 5. Although the situation of path 2 seems to be not a good selection from the UIQLs of neighbor routers, path 2 is a better selection than path 1 because router 9 has shorter UIQL and quickly forwards packets of router 8.

Hence, we should use the UIQLs of adjacent routers (one hop) and more information from neighbors of adjacent routers (two hops) to reduce the misjudgment problem.

The researches of previous 2D NoC only took into account single RF or single SF mechanism. Previous works for 3D NoC exploited hybrid routing algorithms to balance traffic and avoid throttled routers in the chip [18]. However, these schemes were only based on throttling information and should consider congestion situations to effectively transmit packets.

## 3. Traffic-based routing algorithm

In this paper, we propose a traffic-based routing algorithm (TBRA) to dynamically select suitable adaptive routing algorithms according to the network situations and achieve maximize $\overline{T_{PE}}$, minimize $\overline{L_{packet}}$, and $E_{system}$. $\overline{T_{PE}}$ is the average throughput per processor element (PE), $\overline{L_{packet}}$ is the average latency per packet, and $E_{system}$ is the total power consumption of the system.

TBRA consists of a routing function (FTBRF) and a selection function (TBSF). FTBRF combines the features of two adaptive routing algorithms (i.e., WXD and OXD routing

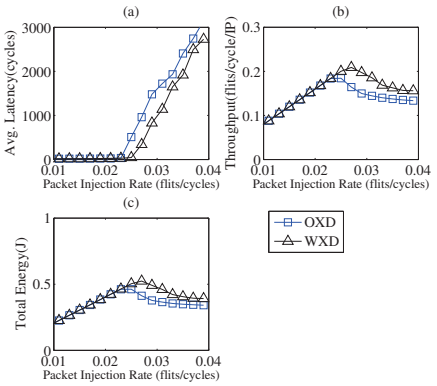



Fig. 2: The performance of OXD and WXD routing algorithms in random traffic.

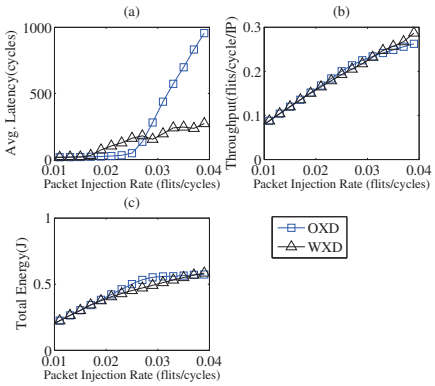



Fig. 3: The performance of OXD and WXD routing algorithms in transpose1 traffic.

algorithms). Most adaptive routing algorithms usually extend from OE and turn mode (i.e, WF routing). OXD and WXD [18] are typical hybrid routing algorithms based on OE [9] and WF [13] routing algorithms, respectively.

TBSF uses a packet forwarding direction table (PFDT) to record the UIQL of each router. The larger UIQL is presented that the router has heavier traffic load and higher routing cost (e.g., larger transmission latency). According to PFDT, TBSF selects the minimum cost routing path (MCRP) to forward packets. MCRP is the least congested routing path for forwarding packets from S to D. Once TBSF obtains the bad routing paths from the original routing algorithm of FTBRA, TBSF sends a feedback to FTBRF. FTBRF dynamically changes to other routing algorithm to obtain better routing paths to transmit the following packets.

Conventional adaptive routing algorithms dynamically determine the proper routing paths and balance traffic in 3D NoC, especially for high traffic load. To observe the features of different adaptive routing algorithms with different traffic loads and traffic patterns in 3D NoC, we efficiently use the features of different routing algorithms (i.e., OXD and WXD) to transmit packets. OXD is based on OE routing, and a router first uses OE routing. If OE routing algorithm cannot be used in a router because of some directional limitations, the router switches to XY routing algorithm to find out other routing paths. If the router still cannot use XY routing algorithm to forward packets, packets are transmitted to below layer by downward routing algorithm. WXD also has the similar routing processes to OXD.

Fig. 2 and Fig. 3 show the features of OXD and WXD including average latency, throughput, and total energy in two most popular synthetic traffic patterns: random traffic and transpose1 traffic [9], [13], [15] by using AccessNoxim [19]. The topology size is set to $8 \times 8 \times 4$, and the packet length is fixed to be 8 flits to neglect the impact of different packet lengths. In random traffic, each source has the same probability to send a packet to each destination. In transpose1 traffic, the coordinates of a destination node ($mesh_x - s.y - 1$, $mesh_y - s.x - 1$, s.z) transpose x coordinate and y coordinate of the source node (s.x, s.y, s.z). $mesh_x$ and $mesh_y$ are the size values of x coordinate and y coordinate from the topology size. Our simulation is from 0.001 to 0.039 flits per cycle with each scale 0.002.

When the PIR is less or equal to 0.02 flits per cycle, we define the situation as 'low' traffic load. On the other hand, when the PIR is higher than 0.02 flits per cycle, the situation is defined as 'high' traffic load.

As shown in Fig. 2 (a), (b), and (c), the performance of WXD is better than that of OXD in random traffic. Since WXD is based on WF, WF has better performance in random traffic [9]. In Fig. 4 (a), (b), and (c), OXD has lower latency and higher throughput when the PIR is less than 0.029 flits per cycle. As the PIR increases, WXD obtains lower average latency and higher throughput. WXD also has lower energy consumption between two routing algorithms.

From Fig. 3 and Fig. 4, we find that OXD and WXD appear different performance in different traffic loads and traffic patterns. Thus, we should choose the suitable routing algorithms according to the network situations to obtain better performance in 3D NoC. Routing algorithm (RA) can be expressed as the functions of $\overline{L_{packet}}$, $\overline{T_{PE}}$, and $E_{system}$ by

$$\begin{aligned} \overline{L_{packet}} &= \frac{\sum_{i=1}^{p_{num}} L_i(RA)}{p_{num}}, \\ \overline{T_{PE}} &= \frac{\sum_{i=1}^{Com\_PE_{num}} T_i(RA)}{Com\_PE_{num} \times Total\_cycles}, \\ E_{system} &= \frac{\sum_{i=1}^{Com\_PE_{num}} E_i(RA)}{Total\_cycles}, \end{aligned} \tag{1}$$

where $L_i(RA)$, $T_i(RA)$, and $E_i(RA)$ are the latency function, throughput function, and energy function of RA, respectively. $p_{num}$ is the number of the transmitted packets. $Com\_PE_{num}$ is the number of received packets by PE

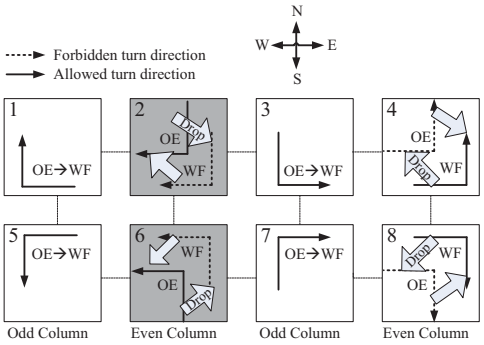



Fig. 4: The switching problem occurs in even columns when RFs of routers switch routing algorithms between OE and WF.

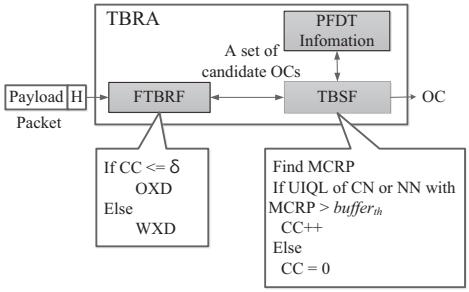



Fig. 5: Trafic-based routing algorithm (TBRA).

which is communicating to other ones. $Total\_cycles$ is the total simulation cycles. In subsection 3.1, we present the proposed FTBRF. TBSF and PFDT are described in subsection 3.2.

## 3.1 Feadback Traffic-based Routing Algorithm (FTBRA)

From Fig. 2 and Fig. 3, OXD appears better performance in low traffic load, and WXD is suitably used in high traffic load. According to the features of Fig. 2 and Fig. 3, a router should select a suitable routing algorithm based on different traffic loads and traffic patterns. However, two issues should be considered. First, we need to obtain the traffic status of a router. Second, we have to decide the switch conditions between the routing algorithms.

Fig. 4 shows the switching problems in even columns when RF of current router changes routing algorithms between OE and WF. The gray blocks show that OE routes packets in some directions but WF does not use these directions. The packets are dropped seriously when routers change from OXD routing to WXD routing, especially for high traffic load. Similarly, RF of current router changes from WXD to OXD, it still encounters the switching problem.

Fig. 5 is the illustration of TBRA. TBRA uses FTBRF

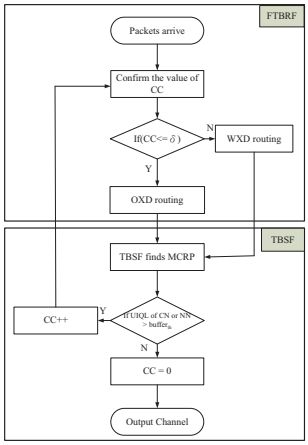


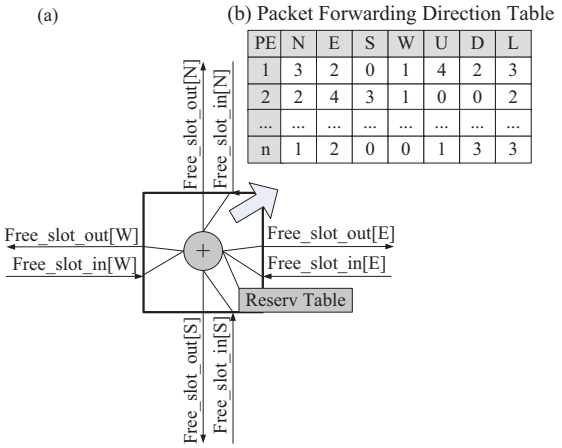

Fig. 6: The flow chart of TBRA.

to forward packets, and TBSF decides MCRP from a set of candidate output ports which are selected by using FTBRF. We use a congestion counter (CC) in the arbiter to show the situations of network congestion. Initially, CC is set to 0. We also use a traffic threshold ($buffer_{th}$) to define the traffic status of a router in TBSF. $buffer_{th}$ presents that the UIQL of a router is X flits. In this paper, the value of X is set to 4. In addition, we use a switching threshold ($\delta$) to switch different routing algorithms of a router. Initially, $\delta$ is set to 0.

FTBRF first checks the value of CC in a router. If CC is less than $\delta$ or equals $\delta$, the router uses OXD routing. Once CC is larger than $\delta$, the router switches to WXD. As shown in Fig. 5, we add an arbiter in each router, and the arbiter detects the buffer information of adjacent routers (the neighbors of the current router (CN)) and neighbors of next routers (the neighbors of next routers (NN)) from PFDT.

If both the UIQLs of CN and NN are less than the threshold ($buffer_{th}$), CC is set to 0. Contrarily, the value of CC increases one and notifies RF. To compare with the value of CC, RF of the current router dynamically uses OXD or WXD, and TBSF selects MCRP according to the buffer information of PFDT to forward packets.

Fig. 6 is the flow chart of TBRA, and it consists of two blocks. The first block is illustrated the operations of FTBRF, and the second block is shown the operations of TBSF. As shown in Fig. 6, a router first checks the value of CC. If CC is less than or equals $\delta$, the router uses OXD routing and TBSF finds MCRP according to the information of PFDT. Once CC is larger than $\delta$, the routing path appears traffic congestion. Hence, the router should use WXD routing to obtain better performance from Fig. 2 and Fig. 3. Our FTBRF dynamically switches the routing algorithms between OXD routing and WXD routing algorithms according to the traffic situations. Then, TSBF selects a proper routing path with MCRP.




| PE | N | E | S | W | U | D | L |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 2 | 0 | 1 | 4 | 2 | 3 |
| 2 | 2 | 4 | 3 | 1 | 0 | 0 | 2 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| n | 1 | 2 | 0 | 0 | 1 | 3 | 3 |

Fig. 7: The packet forwarding direction table (PFDT).

### 3.2 Traffic-based Selection Function (TBSF)

Most of conventional selection strategies only took into account the buffer information of adjacent routers to choose a proper routing path but they resulted in the misjudgment problems (see Fig. 1). However, TBSF finds MCRP according to the information of PFDT. PFDT collects more information to select a more suitable routing path and improve the misjudgment problems.

Fig. 7 is the illustration of PFDT. In Fig. 7 (a), we use the hardware design of NoP [11] to gather the buffer information of neighbors with two hops, and then we record the information of each router on PFDT. The warm-up time of NoC system is 1000 cycles based on default configuration [20]. The warm-up time is to collect PFDT information.

As shown in Fig. 7(b), there are $n$ columns, each column is $\lceil \log_2(buffer_{size}) \rceil + 1$ bits, where $buffer_{size}$ is the input buffer size. The column is denoted the flit number of each output port in a router. Normally, a router has seven output ports in 3D NoC: north port (NP), east port (EP), south port (SP), west port (WP), up port (UP), down port (DP), and local port (LP). Each row is denoted ID of each PE in 3D NoC. The topology size of 3D NoC is set to $dx \times dy \times dz$ ($8 \times 8 \times 4$), the number of PEs is $dxdydz$, and the size of PFDT is $(\lceil \log_2(buffer_{size}) \rceil + 1)ndxdydz$ bits. However, the scalability of PFDT is not good when the topology size increases. In the future, the scalability problems will be improved to use the technology of routing table minimization progresses [21].

When S wants to transmit packets to D, TBSF first checks the buffer information of two hops from PFDT. TBSF calculates a proper routing path with MCRP. The decision function of the best forwarding direction $f_{Best\_Direction}$ is given by $Cost_{current} + Cost_{next\_node}$, where $Cost_{current}$ is the routing cost of the current router and $Cost_{next\_node}$ is the routing cost of next routers.

$Cost_{current}$ and $Cost_{next\_node}$ are calculated based on the buffer information of PFDT. As S sends packets, S uses TBSF to calculate MCRP and alleviate the misjudgment problems based on more accurate network information. However, if there is congestion occurrence with two hops, the selected MCRP may be bad. When CC is larger than $\delta$, TSBF notifies FTBRF to choose another routing algorithm. FTBRF switches OXD routing to WXD routing to find other better routing paths.

## 4. Experiment Results

We use AccessNoxim [19] to show the performance of TBRA, and the simulation parameters refer to literature [22], [23], [24]. AccessNoxim consists of Noxim [20] and Hotspot [25] as well as models the 2D NoC system in literature [26]. The router uses the design of wormhole switching without virtual channel, and $\delta$ is set to 0. To evaluate the performance, two major synthetic traffic patterns are used: random traffic and transpose1 traffic [9], [13], [15]. The primary performance metrics are average latency, throughput, and total energy consumption in the system.

We design two experiments to verify the performance of TBRA. First, we compare the average latency, throughput, and total energy consumption with OXD-NoP in subsection 4.1. OXD is based on OE, and OE is proven to be the best routing algorithm in 2D NoC with non-uniform traffic [9]. NoP is proven to be the best selection function in 2D NoC [11]. In subsection 4.2, we improve the switching problem by adjusting the switching threshold ($\delta$) in different traffic loads and traffic patterns.

### 4.1 The Performance of TBRA

As shown in Fig. 8(a), (b), and (c), the performance (average latency, throughput, and total energy) of TBRA sharply decreases as PIR increases. TBRA uses OXD routing in light traffic load and then switches to WXD routing in high traffic load. When routers switch to other routing algorithm in even columns, different directional limitations between OE and WF result in performance degradation in high traffic load.

Fig. 9 divides into two parts (the PIR is below 0.027 flits per cycle and above 0.027 flits per cycle) to discuss the performance of TBRA. When the PIR is below 0.027 flits per cycle, TBRA has a little higher average latency and higher total energy than OXD-NoP. The reason is that TBRA uses more complex hardware design to collect more network information. As the PIR is above 0.027 flits per cycle, TBRA improves 17.62% average latency and 0.57% throughput. However, TBRA has a little higher total energy (0.27%) because of the complex hardware design.

### 4.2 The switch threshold of FTBRA

We adjust the switching thresholds ($\delta$) to improve the switching problem. Fig. 10 and Fig. 11 show the impact of different switching thresholds ($\delta$) for TBRA performance in

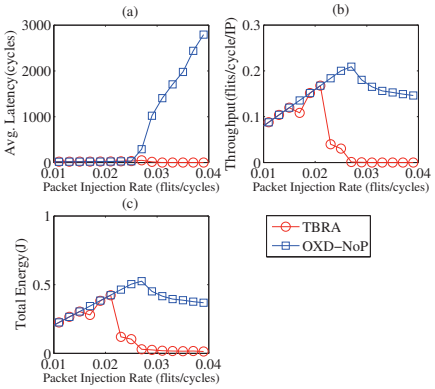



Fig. 8: The performance of TBRA and OXD-NoP in random traffic.

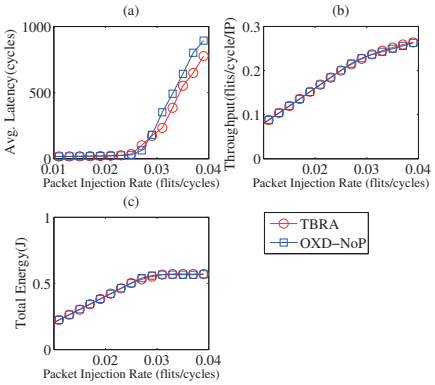



Fig. 9: The performance of TBRA and OXD-NoP in transpose1 traffic.

random traffic and transpose1 traffic. The $i_{th}$-TBRA denotes that the value of CC is larger than $i$ to change routing algorithm from OXD routing to WXD routing. The larger value of CC is present that the duration of traffic congestion has continued CC cycles.

Our TBRA uses the features of OXD and WXD to improve the system performance. In Fig. 10, we find that the larger switching threshold reduces the number of switch algorithm from OXD routing to WXD routing. When we use a larger $\delta$, the performance of TBRA approaches that of OXD-NOP.

From Fig. 4, if two conditions are satisfied simultaneously, the switching problem occurs. One is that the current router switches to other routing algorithm. The other is that the switching position appears in even columns. From Fig. 10, we find that the $10_{th}$-TBRA appears unstable states. When the PIRs are equal to 0.027, 0.033 and 0.035 flits per cycle, we find that the duration of traffic congestion is larger than 10 cycles, and many hotspots appear in even columns. In addition, a router easily changes from OXD routing to WXD routing due to the smaller threshold (i.e., $\delta$=10). Hence,

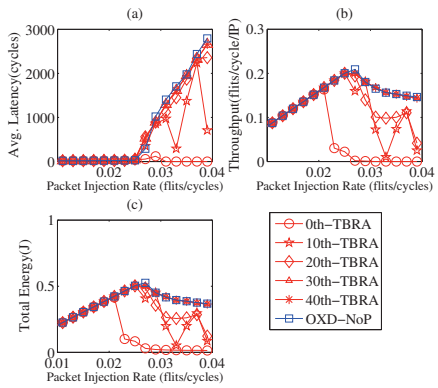



Fig. 10: The performance of TBRA with different switching threshold ($\delta$) in random traffic.

the switching problem occurs, and the $10_{th}$-TBRA suffers serious throughput degradation.

The $0_{th}$-TBRA and the $10_{th}$-TBRA are not the proper switch thresholds because most duration of traffic congestion is larger than 10 cycles. When the PIR is below 0.029 flits per cycle, the $30_{th}$-TBRA has a little higher average latency and higher total energy than OXD-NoP. The reason is that TBRA uses more complex hardware design to collect more network information. When the PIR is above 0.029 flits per cycle, the $30_{th}$-TBRA improves 6.11% average latency. However, the $30_{th}$-TBRA has a little lower throughput (-0.17%) and consumes lower total energy (0.16%). As $\delta$ equals 40, the performance of the $40_{th}$-TBRA is the same as the $30_{th}$-TBRA. Fig. 10 shows that the better value of $\delta$ is set to 30 in random traffic.

In Fig. 11, we can find that the $0_{th}$-TBRA has the lowest average latency in transpose1 traffic, and the performance of the $30_{th}$-TBRA approaches to that of OXD-NoP. Summarily, TBRA exploits OXD routing in light traffic load and then switches to WXD in high traffic load. It obtains better performance than that of OXD-NoP in transpose1 traffic with high traffic load. Although TBRA with $\delta = 0$ has worse performance in random traffic, the traffic distribution of the chip rarely appears random traffic in realistic 3D NoC. We also improve the performance in random traffic by adjusting the switching threshold. The better value of $\delta$ is set to 30 in random traffic, and $\delta = 0$ is a better choice in transpose1 traffic.

## 5. Conclusion

In this paper, we propose a TBRA to combine FTBRF with TBSF. FTBRF dynamically uses OXD or WXD routing depending on traffic load. TBSF exploits more accurate network information to improve the network congestion problems.

We use TBRA to obtain the advantages of hybrid routing algorithms and address better performance in different

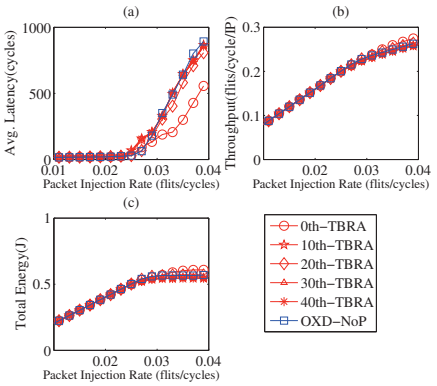



Fig. 11: The performance of TBRA with different switching threshold ($\delta$) in transpose1 traffic.

traffic loads and traffic patterns. According to the simulation results, we find that TBRA show the better performance in transpose 1 traffic with high traffic load. We adjust switching threshold to improve the switching problem. The better value of $\delta$ is set to 30 in random traffic, and $\delta = 0$ is a better choice in transpose 1 traffic.

Therefore, the designers use different routing algorithms and different switching thresholds to achieve better performance in different traffic loads and traffic patterns. TBRA is a better hybrid routing algorithm for enhancing performance in 3D NoC.

## Acknowledgment

The authors would like to thank the Ministry of Science and Technology of the Republic of China, Taiwan, for financially supporting this research under Contract No. MOST 104-2221-E-005-011 and 104-2221-E-146-003-. We also thank to Access Lab for sharing the code of Access-Noxim simulator.

## References

[1] G. E. Moore, *Cramming more components onto integrated circuits,* IEEE Solid-State Circuits Newsletter, vol. 11, pp. 33-35, September 2006.

[2] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, *Cluster-based Topologies for 3D Networks-on-Chip Using Advanced Interlayer Bus Architecture,* Journal of Computer and System Sciences, vol. 79, pp. 475-491, June 2013.

[3] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, *3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration,* Proceedings of IEEE, vol. 89, pp. 602-633, May 2001.

[4] E. Beyne, "3D system integration technologies,*Int. on Symp. VLSI Technology, Systems, and Applications,* pp. 1-9, April 2006.

[5] J. Q. Lu, K. Rose, and S. Vitkavage, *3D integration: Why, what, who, when?,* Future Fab Int., no. 23, pp.25 -27 2007.

[6] A. Sheibanyrad, F. Petrot, and A. Jantsch, *3D Integration for NoC-based SoC Architectures,* Springer, Nov 2010, ISBN 978-1-4419-7617-8.

[7] P. Mohapatra, *Wormhole routing techniue for directily connected multicomputer systems,* ACM Computing Surveys, vol. 30, pp. 374-410, September. 1998.

[8] Z. Wang, H. Ligang, G. Jinhui, and W. Wuchen, *Comparison Research between XY and Odd-Even Routing Algorithm of a 2-Dimension 3X3 Mesh Topology Network-on-Chip,* Global Congress Intelligent Systems, vol. 3, pp. 329-333, May 2009.

[9] G. M. Chiu, *The odd-even turn model for adaptive routing,* IEEE Transactions on Parallel and Distributed Systems, vol. 11, pp. 729-738, July 2000.

[10] H. Gu, J. Xu, K. Wang, and H. Morton, *A new distributed congestion control mechanism for networks on chip,* Telecommunication Systems, vol. 44, pp. 321-331, August 2010.

[11] G. Ascia, V. Catania, M. Palesi, and D. Patti, *Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip,* IEEE Transaction on Computers, vol. 57,In this paper, we propose a novel traffic-based routing algorithm, TBRA, to achieve better performance. TBRA combines some advantages from different routing algorithms and selects adaptive routing algorithms according to the traffic load. Furthermore, it also collects two hops data of networks to determine the better routing path. pp.809-920, June. 2008.

[12] M. Li, Q. Zeng, and W. Jone, *DyXY - a proximity congestionaware deadlock-free dynamic routing method for network on chip*, ACM/IEEE 43rd Design Automation Conference, pp. 849-852, 2006.

[13] C. J. Glass, and L. M. Ni, *The Turn Model for Adaptive Routing,* The Annual 19th International Symposium on Computer Architecture, pp. 278-287, 1992 .

[14] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi, *BARP-A Dynamic Routing Protocol for Balanced Distribution of Traffic in NoCs to Avoid Congestion,* Design, Automation, and Test in Europe Conference (DATE), pp. 1408-1413, Mar 2008.

[15] Jingcao Hu, and R. Marculescu, *DyAD smart routing for networks on chip,* Proceedings of 41th Design Automation Conference, pp. 260-263, July 2004.

[16] F. A. Samman, T. Hollstein, and M. Glesner, *Runtime Contention and Bandwidth-Aware Adaptive Routing Selection Strategies for Networks-on-Chip*IEEE Transaction on Parallel and Distributed Systems, vol. 24, pp. 1411-1421, July 2013.

[17] S. Ma, J. E. Natalie, W. Zhiying, M. Lai, and L. Huang, *Holistic Routing Algorithm Design to Support Workload Consolidation in NoCs* IEEE Transactions on Computers, vol. 63, pp.529-542, March 2014.

[18] K. C. Chen, S. Y. Lin, H. S. Hung, and A. Y. Wu., *Topology-Aware Adaptive Routing for NonStationary Irregular Mesh in Throttled 3D NoC Systems,* IEEE Transaction on Parallel and Distributed Systems, vol. 24, pp. 2109-2120, October 2013.

[19] K. Y. Jheng, C. H. Chao, H. Y. Wang, and A. Y. Wu, *Traffic-thermal mutual-coupling co-simulation platform for three-dimensional Network-on-Chip,* 2010 International Symposium on VLSI Design Automation and Test, pp. 135-138, April 2010.

[20] Sourceforge.net, Noxim: Network-on-chip simulator, 2008. [Online]. Available: http://noxim.sourceforge.net

[21] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, *Routing Table Minimization for Irregular Mesh NoCs,* Design, Automation & Test in Europe Conference & Exhibition, pp. 1-6, April 2007.

[22] R. S. Ramanujam, and Bill Lin, *Destination-based adaptive routing on 2D mesh networks,* 2010 ACM/IEEE Symposium Architectures for Networking and Communications Systems, pp. 1-12, October 2010.

[23] P. Lotfi-Kamran, A. M. Rahmani, M. Daneshtalab, and A. Afzali-Kusha *EDXY - A low cost congestion-aware routing algorithm for network-on-chips,* Journal of Systems Architecture, vol. 56, pp. 256-264, July 2010.

[24] U. Y. Ogras and R. Marculescu, *Prediction-based Flow Control for Network-on-Chip Traffic,* 2006 43rd ACM/IEEE Design Automation Conference, pp. 839-844, 2006

[25] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, *HotSpot: A compact thermal modeling methodology for early-stage VLSI design,* IEEE Transaction on Very Large Scale Integration System, vol. 14, pp.501-513, May 2006.

[26] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, *A 5-GHz mesh interconnect for a teraflops processor,* IEEE Micro, vol. 27, pp. 51-61, Jan. 2007.

ISBN: 1-60132-433-2, CSREA Press ©

# SESSION

# ALGORITHMS AND APPLICATIONS: CONTROL SYSTEMS + SECURITY AND RELATED ISSUES

## Chair(s)

**TBA**

ISBN: 1-60132-433-2, CSREA Press ©

ISBN: 1-60132-433-2, CSREA Press ©

# An Assessment of the Environmental Noise Effects on the Performance of Configurable based ROPUFs

**Fathi Amsaad, Chayanika Roychaudhuri, Atul Prasad, and Mohammed Niamat**
Electrical Engineering and Computer Science, University of Toledo, Toledo, OH
Email: {famsaad, croychaudhuri, aprasad }@rockets.utoledo.edu, mniamat@utoledo.edu

***Abstract*— Silicon Physical Unclonable Functions (SPUFs) are security primitives that are embodied in a chip and exploit the manufacturing process variations of the Integrated Circuits (ICs) to extract unique secrets for chip authentication and cryptographic key generations. Due to their simple implementation and high performance, Ring Oscillator PUFs (ROPUFs) are one of the most suitable security solutions for silicon technology devices including ASICs and FPGAs. As far as our knowledge goes, there is no comprehensive research that assesses the impact of varying environmental conditions on the performance of controlled–inverter Configurable ROPUFs (c-ROPUF). In this paper, we use our previously proposed c-ROPUF design to analyze RO sample frequencies that are extracted from five Spartan 3E FPGAs (90 nm) at five different temperatures & voltages. The experimental results show that RO frequencies follow a fixed pattern which shows that environmental variation effects are uniformly distributed on the individual ROs throughout the FPGA chips. However, RO frequencies are exposed to high bit flips percentage due to voltage variations compared to temperature variations.**

***Index Terms*— FPGAs, hardware security, temperature variations, voltage variations, ROPUF performance.**

## I. INTRODUCTION

Owing to their re-programmability and higher flexibility to offer great amount of hardware resources, including SRAM memory cells, DSPs, and dedicated primitives, FPGAs have recently become key elements in the embedded systems and life critical applications. The inability to fabricate identical silicon chip (100 % identical) introduces a random delay that is inherited by the manufactured ICs in each chip and is known as manufacturing process variations. SPUFs take advantage of the manufacturing process variations to authenticate the chips.

ROPUFs are one-way probabilistic functions that exploit process variations in order to map challenge 'c' of a physical entity to its corresponding response 'r' [1]. For example, when the challenge 'c' is applied, a pair of identically instantiated ROs ($RO_a$, $RO_b$) is selected from two locations on the same chip for comparison. Process variation's delay ($PV_a$ , $PV_b$) for a pair of ring oscillators ($RO_a$, $RO_{b)}$ that are mapped on different areas of the same $chip_i$ ($1 \leq i \leq m$), $m$ is the number of chips under test, will slightly differ from each other. Based on these differences, ROPUF generates two frequencies $f_a$, $f_b$ for $RO_a$, $RO_b$ respectively, where $1 \leq a \leq n$ , $1 \leq b \leq n$, and n is the number of the ROs mapped on a certain chip. The two frequencies are compared and one response-bit, r = '1' or '0', is generated based on the following formula:

$$r = \begin{cases} 1, \ if \ fa \geq fb \\ 0, \ Otherwise \end{cases} \quad (1)$$

On account of the randomness property of the process variations in a ROPUF design, the generated frequencies are inherently unique for the individual ROs. This facilitates a high unpredictability in the generated RO frequencies that ensures greater unclonablity in the response bits extracted using these frequencies. In addition, ROPUF is a tamper-proof design that can detect on-chip physical tampering attempts. Physical tampering permanently affects the process variation of the individual ROs that are mapped in the tampered area of a silicon chip. Hence, when mapped on that tampered area, ROPUF design generates different response bits which indicate a tampering attempt in that particular area of the chip. As shown in equation 1, in order to generate one response bit, a pair of RO frequencies (two RO frequencies) is needed. For a small CRPs space, it is possible that an adversary tries all challenges and knows the corresponding responses.

Configurable ROPUF was first introduced by Maiti in [2] to basically enhance the reliability of original ROPUF proposed by Devadas in [1]. To expand the CRPs space, an improved version of Maiti's configurable ROPUF, namely c-ROPUF, is proposed by Amsaad in [3]; which efficiently utilizes FPGA logic to create more number of ROs (double) within the same area (single CLB) and uses controlled inverter PXOR gates to control the RO frequencies. An increased CRPs space leads to a robust design against the modeling and machine learning's attacks [4].

Environmental variations temporarily change process variations around the individual RO loop which cause noisy frequency being generated by the individual ROs. Thus, ROPUF suffers from instability in the generated response bits (bit flips), which in turn decreases the ability of ROPUF to generate the same response bits at varying operating conditions (ROPUF reliability) [1, 2, 5]. In addition, a higher bit flips percentage reduces the percentage of CRPs (RO frequency pairs) that can be used to generate unique responses; this negatively impacts the overall uniqueness of ROPUF. The performance of ROPUF in terms of uniqueness and reliability of the generated response bits is negatively affected by the environmental variations. Thus, the important question here is, how to properly study these effects. In this paper, we investigate this problem by exploring the impact of environmental variations on the generated ROs frequencies using Spartan-3E FPGAs that are based on 90nm nanometer technologies. The design is implemented on the entire area of 5 FPGAs.

Secondly, the correlation between the generated average sample frequencies and the environmental variations i.e. voltage, and temperature variations is determined. The average frequencies of the ROs are found to be inversely proportional to the applied temperature and directly proportional to the applied voltages. Hence, an algorithm to predict the ROPUF environmental noise is proposed to improve the overall performance of ROPUF.

## II. BACKGROUND

Physically Unclonable Functions (PUF) is a modern hardware primitive for the security of silicon technology chips including FPGAs and ASICs. Based on their randomness property, PUFs can be categorized into "explicitly" and "implicitly introduced" randomness [6, 7]. The explicitly introduced PUFs, like optical PUFs and protective coating PUFs, are known as the non-intrinsic PUFs and their randomness is internally evaluated using special measuring equipment [6, 8]. Protective coating PUF was proposed by Tuyls et al which comprises of a coating and randomly spread dielectric materials [9]. The optical PUFs proposed by Pappu et al. in [10] exploit the unique pattern of a speckle that ascends when applying laser against a transparent item to generate highly random PUF signature. On the other hand, the implicitly introduced PUFs are mainly Silicon (S) PUFs that use the intrinsic manufacturing process variations to produce secret keys. Considering its simple implementation and high performance for both ASICs and FPGAs, a Ring Oscillator PUF (ROPUF) is one of the most popular silicon PUFs.

Fabrication of silicon chips are done over a submicron, and results in nearly identical ICs that slightly differ in the effective structural parameters, such as channel length L, channel width W, transistor doping concentration NA, and the oxide thickness Tox [11]. A relatively small processing delay that differs from one IC to another is inherited by these ICs; which is also known as the static process variation and is typically influenced by the effective transistor gate length (L) parameter [11]. Silicon (S) PUFs including Arbiters (A) PUFs and (RO) PUFs can be easy to implement and evaluate on FPGA chips and their output is difficult to clone or predict. ROPUFs exploit the uniqueness of the process variation in order to obtain a unique cryptographic key for security applications such as Intellectual Property (IP) Protection, Smart Grid, and on-chip authentication. Fig. 1 shows the original ROPUF design that was introduced by Suh, G. E., and Devadas in [2]. To obtain a bit response from the ROPUF circuit, a frequency pair of two identically instantiated ROs is compared. Neglecting the noise effect, the measured process variations is unique for each RO.

An Arbiter (A) PUF is a silicon based PUF that was introduced in 2004 by Lee et al [12]. Fig. 2 shows the basic structure of APUF. As shown in the figure, APUF uses switch-box structure to create a race between two delay paths with an arbiter at the end of the two paths. Furthermore, the arbiter is connected at the end of the design as a D-latch flip-flop (other types of flip flops might be used), with one input connected to the clock signal and the other connected to the data input so as to select the wining signal, which is the one that reaches the arbiter first.

In 2007 G. Suh and S. Devadas proposed a 1-out-of-k technique to enhance ROPUF reliability [1, 2]. This technique basically selects an RO pair that has the maximum frequency difference among k-RO pairs. The downside of this technique is

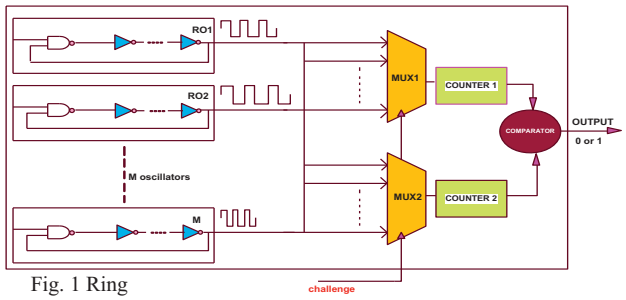



Fig. 1 Ring Oscillator based PUF circuit [2].

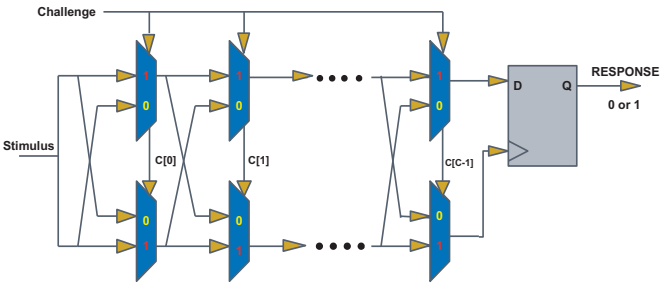



Fig. 2 An arbiter PUF delay design [7].

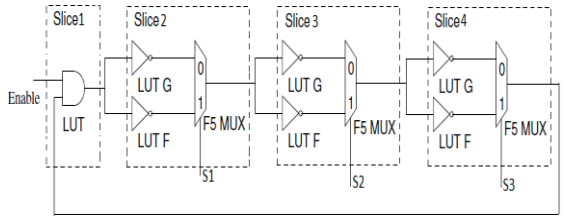


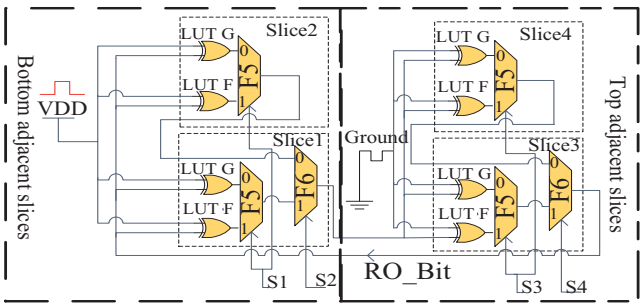

Fig. 3 Configurable ROPUF in One CLB proposed in [1].

the area inefficiency where k times more area is sacrificed when implementing it on silicon chips.

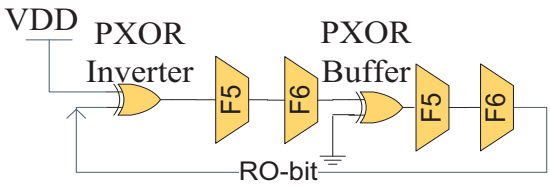
To overcome this drawback, in 2009 Maiti [2] introduced an FPGA-based configurable ROPUF in a single Configurable Logic Block (CLB) using Spartan-3E family.

Fig. 3 shows the basic structure of Maiti's configurable RO design inside one CLB which uses S1, S2, and S3 to configure one RO among eight different ones. Each RO has a different frequency based on the process variation delay of its dedicated logic. To satisfy the identical routing requirements, RO pairs are selected from different CLBs with the same configuration. In his design, Maiti used the redundancy approach technique (proposed by Devadas) to select the RO pair that has the highest frequency difference among a group of eight configurable RO pairs that are mapped on two adjacent CLBs to enhance the reliability of ROPUF. The main advantage Maiti's configurable ROPUF design is that it can be implemented on the same area as Devadas's design, but with more reliable CRPs.

Fig. 4 (a) shows a controlled configurable ROPUF inside one CLB proposed by Amsaad in [3].This design makes an efficient use of the internal routings to connect the LUTs and the dedicated multiplexers with fixed routing delay. As shown in Fig. 4 (a), using selection lines S1 to S4, the proposed design allows instantiation of 16 RO paths.




(a)




(b)

Fig. 4 configurable ROPUF design proposed in [3]: (a) 16 ROs in a single CLB, (b) One RO loop.

Restricting the design to one CLB is very important in order to exploit the unique behavior of the internal CLB routings that connect between the dedicated logic with fixed delay.

Four PXORs are implemented at slice1 and slice2 which acts as controlled inverters (CNOT gates). To accomplish that, each LUT in Slice 1 & Slice 2 is first configured as two-input PXORs. One input of each PXOR is then connected to logic 1 and the other input is connected to the output of the active RO. A pair of these PXORs is then connected to its corresponding F5MUX. FiMUX from slice1 and slice 2 is programmed as F6MUX to connect its corresponding F5MUXs in these slices. In order to bypass the RO-bit signal, the output of bottom F6MUX is connected to all four LUTs at top slices, slice 3 and slice 4. These LUTs are programmed as buffers by configuring each of them to the PXORs, then connecting one of their inputs to logic 0 (ground) and the other input to the bottom F6MUX. Each two of these PXORs are connected to its corresponding F5MUX in the same slice. F5MUXs at slice 2 and 3 are connected to the top configured F6MUX in slice 3. Configrable ROPUF output is driven from top F6MUX and fed back to PXORs at the top two slices (slice 1 and slice 2) so as to control the oscillation of the active RO.

As illustrated in Fig. 4 (b), to control the output of each RO, two PXOR gates are used as an inverter or buffer in each RO-loop. Due to the limitations in possible number of CRPs space in the original ROPUF design, secret keys might not be enough to satisfy the unpredictability and resistance to modeling attacks. Configurable ROPUF is proposed in [3, 7] to provide large number of ring oscillators in a small area of the chip which strengthens the generated secrets, increases the unpredictability and ensures high resilience of ROPUF design against machine learning and molding attacks.

As far as our knowledge goes, performance issues of c-ROPUF frequencies under different conditions were not addressed. In the following sections, we show the experimental setups and analyze the frequency of our configurable ROPUF in order to validate its performance under different conditions. In addition, critical quality factors of the configurable ROPUF are explored under different temperature variations to justify the overall performance of the design.

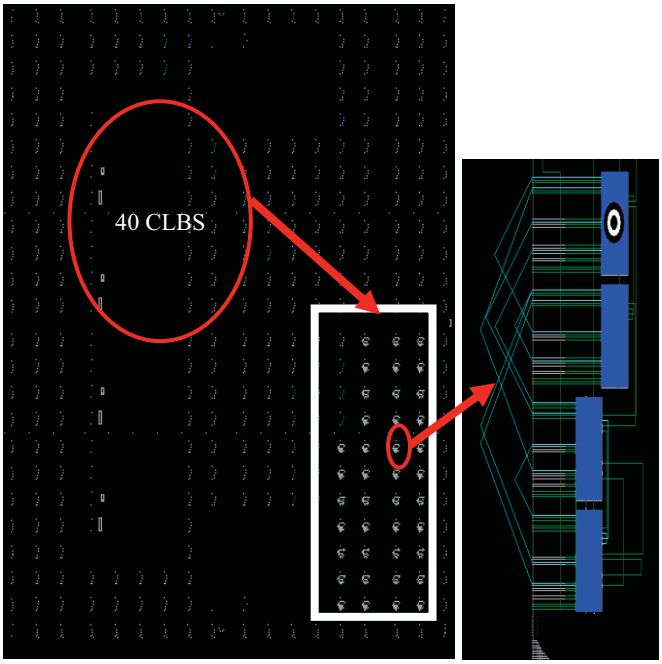
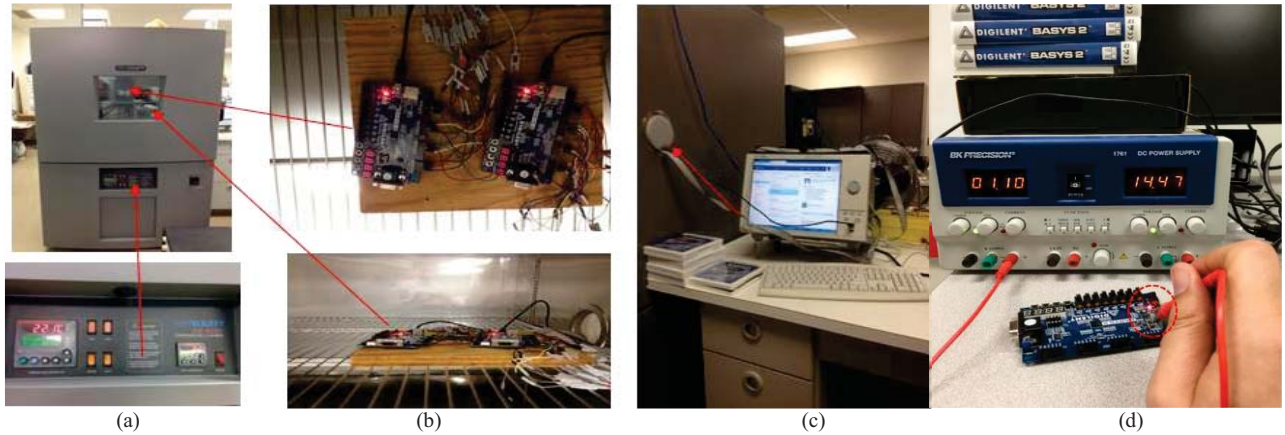



Fig. 5 A hard macro of 40 CLBs in the bottom FPGA area mapped in the bottom right FPGA area [3].

## III. EXPERIMENTAL SETUP

As shown in Fig. 6, the design is first implemented inside a single CLB comprising 16 different ROs with fixed internal routings. Fixed CLB routings eliminate possible noise from the dynamic routing due to the use of external routings and guarantees that the generated frequencies depend highly upon the random manufacturing process variations of the individual ROs.

The design is implemented on five Spartan 3E FPGAs (90nm), each consisting of 240 CLBs. Each FPGA is divided into six equal regions with 40 CLBs and 640 ROs per region. Using Xilinx ISE CAD Tools, a hard macro procedure is performed on a single CLB and duplicated on the entire region (40 CLBs). As a result, six bit-stream files (one file per region) are obtained. These files guarantee faster mapping of the design and ensure that all ROs are identically mapped on each FPGA region using internal CLBs routings with a fixed delay. Data samples are collected from the six regions of each chip. As seen in Fig. 6, a 1000 series Test-Equity chamber is used with an Agilent 18601 logic Analyzer to collect sample frequencies at different temperatures and DC voltage supply. As shown in Fig. 6(a), the temperature chamber contains racks where FPGAs are placed. The environmental temperature for FPGAs under test is controlled by a temperature control unit. This unit is embedded in the chamber and has the ability to change the environmental temperature inside the chamber. FPGAs are placed in the chamber and the environmental temperature is adjusted to a desired placed in the temperature chamber each time and data samples are collected using the logic analyzer. Data samples are collected from the entire area of the FPGA (240 CLBs) at $0^{o}$C, $25^{o}$C, $50^{o}$C and $75^{o}$C. As shown in Fig. 6 (c), Diligent-Adept software is used to download the bit stream file on each FPGA region. After the FPGA region is configured, it is connected to a 16-bit data bus of 16801 Agilent Logic Analyzer. The Logic-Analyzer can accurately measure data samples and observe the non-harmonic frequencies of the individual ROs. Fig. 6 (d), shows one of the FPGAs tested under different voltages using a DC power supply.




(a) (b) (c) (d)

Fig. 6 Experimental equipment: (a) temperature control panel, (b) two Basys-2 Spartan-3E FPGAs under test inside the temperature chamber, (c) Agilent 18601A logic analyzer connected to FPGAs inside the temperature chamber, and (d) Basys-2 Spartan-3E FPGAs under test using DC power supply

## IV. RESULTS AND ANALYSIS

Temperature and voltage variation experiments were conducted using the above experimental equipment as seen in Fig.6.

Table 1 shows the average regional frequency for each $k^{th}$ region in an $i^{th}$ FPGA (six regions per FPGA) at certain temperature and supply voltages. In the Table, average FPGA regional frequency at five different temperatures with a $25^0$C difference between each temperatures, and five different supply voltages with a 1V difference between each applied voltage is calculated. To perform that, a total of 150 data sheets for all the 5 tested FPGAs at different temperatures are collected by our logic analyzer for analysis. For accuracy, 10 data samples ( $Sf_{i,j,k,l}$ ) from each data sheet are used to estimate the average frequency ($Avg.RO_{(i,j,k,l)}$) for each ring oscillator using the formula as follows [5, 13]:

$$Avg.RO_{(i,j,k)} = \frac{1}{10}\sum_{o=1}^{10} S.f_{(i,j,k,l)} \tag{2}$$

where $Sf_{i,j,k,l}$ represents the $l^{th}$ frequency sample, ($1 \le l \le 10$) of the $k^{th}$ RO,($1 \le k \le 640$) in the $j^{th}$ FPGA region, ($1 \le j \le 6$) of the $i^{th}$ FPGA, ($1 \le i \le 5$) and evaluated as follows :

$$S.f_{(i,j,k,l)} = \left(RO_{CC_{(i,j,k,l)}} / \ Ref_{CC_{(i,j,k,l)}}\right) \times \ 50 \ MHz \tag{3}$$

$Refcc_{i,j,k,l}$ is the default Spartan 3E-FPGA clock which is used as a reference clock, $RO_{CC_{i,j,k,l}}$ is the number of clock cycles (data sample) of the active RO. The average regional frequency is calculated for each FPGA region at certain temperature T=t as shown in equation 4:

$$Avg.\text{Re}\,gional_{(i,j)_{\downarrow T=t}} = \sum_{k=1}^{640} Avg.RO_{i,j,k} \tag{4}$$

As shown in Table 1, average regional frequencies are inversely proportional to the applied temperature at $0^o$C, $25^o$C, $50^o$C and $75^o$C. However, average regional is directly proportional to the applied temperature frequencies at 1V, 1.1V, 1.2V, 1.3V and 1.4V. This can lead to bit flips occurrence at bit positions and affect the response bits. As a result, the following formulas that represent the relation between the regional frequency at certain temperatures 't1' and 't2', and voltages 'v1' and 'v2' can be written as follows:

$$Avg.\text{Re}\,gional_{(i,j)_{\downarrow T=t1}} \alpha \frac{1}{Avg.\text{Re}\,gional_{(i,j)_{\downarrow T=t2}}} \tag{5}$$

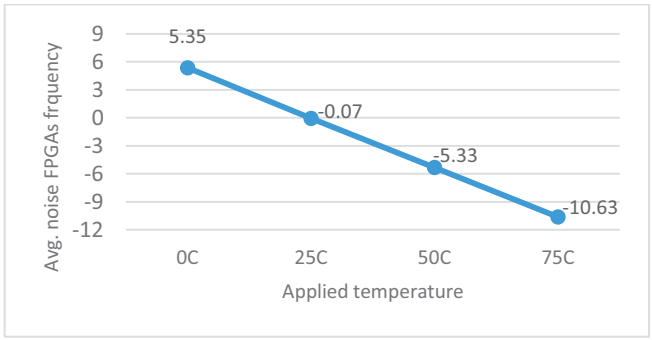
$$Avg.\text{Re}\,gional_{(i,j)\downarrow V=v_2} \alpha Avg.\text{Re}\,gional_{(i,j)\downarrow V=v_1} \tag{6}$$

where t1<t2 (t2-t1=25°C) and v1<v2 (v2-v1=1V).

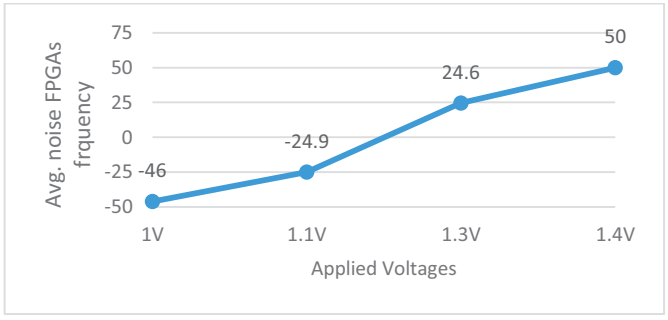
Therefore, noise frequency is calculated using the data in Table 1 with the help of the following equation:

$$Noise\_Frquency = Avg.\text{Re}\,gional_{\downarrow T=t,V=v} - Avg.\text{Re}\,gional_{\downarrow T=RT,V=1.2v} \tag{7}$$

As shown in Fig. 7 (a), at $0^o$C, average regional frequency increases by 5.35 MHz, at $25^o$C the frequency decreases by 5.33 MHz and at $75^o$C the frequency decreases by 10.63 MHz with an average of 5.3 MHz (noise frequency) for $25^o$C change in the applied temperature.




(a)




(b)

Fig. 7 Average noise frequencies on five FPGAs : (a) due to temperature variations (b) due to voltage variations

Table .1 Average RO frequencies of the tested FPGAs varying temperatures and supply voltages.

| FPGA Regions | | Applied Temperature (Celsius) | | | | | Applied voltage (Volts) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0°C | Rt °C | 25 °C | 50 °C | 75 °C | 1V | 1.1V | 1.2V | 1.3V | 1.4V |
| Bottom | Left | 279.66 | 274.08 | 273.96 | 269.02 | 264.42 | 228.4 | 249 | 274.1 | 298.6 | 324.9 |
| | Middle | 281.2 | 275.94 | 275.64 | 270.52 | 266.14 | 229.9 | 250.1 | 275.9 | 300.2 | 325.9 |
| | Right | 278.56 | 273.62 | 272.96 | 267.76 | 263.8 | 227.2 | 249.3 | 273.6 | 297.6 | 323.3 |
| Top | Left | 279.88 | 274.24 | 274.74 | 269.18 | 264.42 | 228.6 | 249.6 | 274.2 | 299.3 | 324.6 |
| | Middle | 281.04 | 276.36 | 275.78 | 270.8 | 266.64 | 230.2 | 250.3 | 276.4 | 300.4 | 325.8 |
| | Right | 279.7 | 274.12 | 274.86 | 266.64 | 264.64 | 228.1 | 250.2 | 274.1 | 299.5 | 324.4 |
| Average Regional Frequency | | 280.01 | 274.73 | 274.66 | 269.33 | 265.01 | 228.7 | 249.8 | 274.7 | 299.3 | 324.7 |

As far as the voltage variation noise is concerned, the noise frequency increases by 50 MHz for every 1V change, and it increases up to 24.6 MHz for 1.4V and 1.3V respectively as shown in Fig. 7 (b). However, it decreases by 24.9 MHz and by 46 MHz for 1.1V and 1.0V respectively.

In addition, environmental temperature variations and noisy oscillators can cause the response bits to flip. For a certain challenge, reliability can be evaluated by calculating Hamming Distances (HDs) between $rs_{,rt}$ the $t^{th}$ response bit from a $chip_i$ at room temperature (normal operating condition), and $rs_{,t}$ the $t^{th}$ response bit from the same chip at varying operating condition such as varying temperature or voltage condition. The ideal bit flips percentage for response bits that are generated from the same chip should be 0%.

The percentage of bit flips is calculated as follows [5, 14]:

$$Bit\_flip = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{HD(r_{s,rt}, r_{s,t})}{n} \times 100\% \quad (8)$$

Fig. 8 shows that the probability of bit flips. It is shown in the figure that as the frequency difference increases the probability of bit flips decreases and vice versa. From the figure, it is found that the bit flip occurs most frequently when the maximum frequency difference is 1 MHz and 1.5 MHz on account of temperature and voltage variations respectively.

Fig.9 shows the effect of environmental temperature variations on all FPGAs. It is observed that for every FPGA, the average regional frequencies decrease with increase in temperature and vice versa. To verify this correlation between sample RO frequencies and the environmental conditions, IBM-SSPS is used. Fig.10 (a) shows a strong negative correlation with correlation factor of r = -0.99. This means for every 1°C increase in the temperature, the average RO frequency will change by 0.2 MHz. However, Fig. 10 (b) shows a strong positive correlation with correlation factor of r = + 0.99. This means for every 1V increase in the voltage variations, the average RO frequency will increase by $2.42 \times 10^2$ MHz. Although the regional RO frequencies has high increment with respect to the voltage variations when compared to temperature variations, the frequencies follow a fixed pattern which implies the voltage variation effect is also uniformly distributed throughout the FPGA chip.

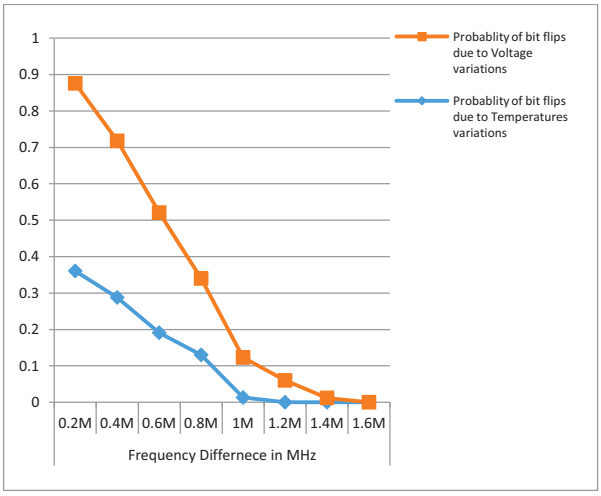



Fig.8 Bit flip probability vs frequency difference (MHz)

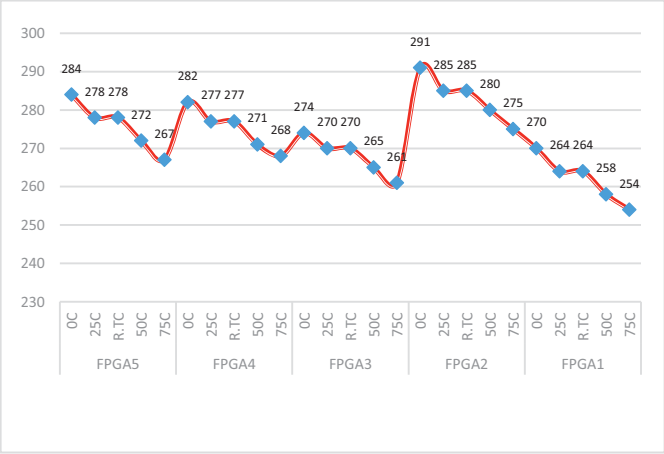
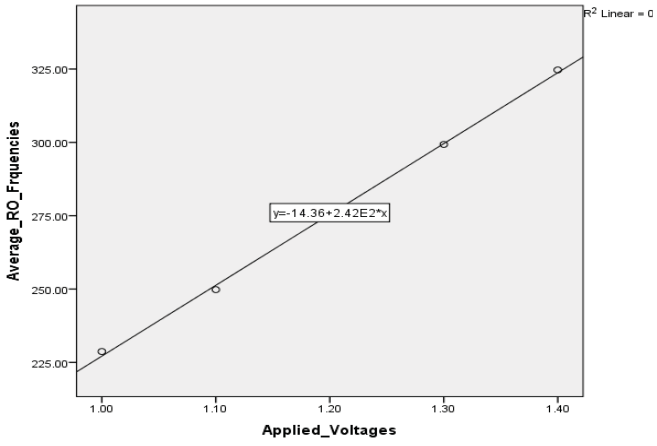
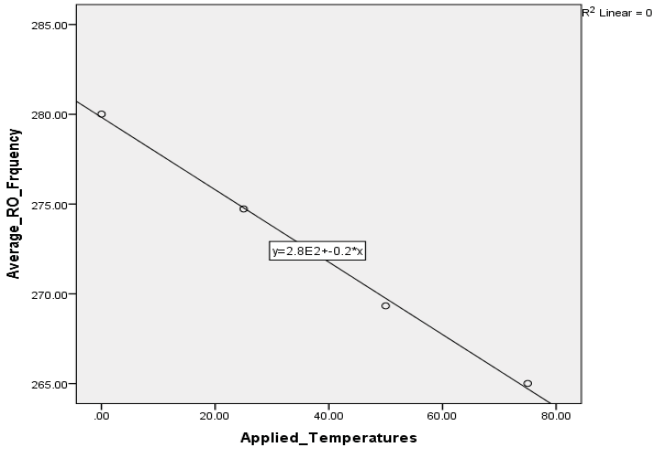



Fig. 9. Effects of temperature variations on all FPGAs




(a)




(b)

Fig. 10 The correlation between average RO frequencies of 5 FPGAs: (a) temperature variations, (b) voltage variations

## V. CONCLUSIONS

An area efficient controlled inverter configurable ROPUF is implemented on Spartan 3E FPGAs using Programmable XOR gates. The RO frequencies are recorded at six different regions on each FPGA under different temperature variations including room temperature ($21^0$ C), $0^0$ C, $25^0$ C, $50^0$ C and $75^0$ C. Regional RO frequencies are also recorded at the same regions on each FPGA for different voltage variations including 1.2V (normal), 1V, 1.1V, 1.3V, and 1.4V. It is observed that the average RO regional frequencies of the tested chips increase with a decrease in temperature and vice versa. However, it is noticed that the average RO regional frequencies of the tested chips increase with an increase in voltage and vice versa. Our result shows that the most bit flips occur when the maximum frequency difference is 1 MHz and 1.5 MHz on account of temperature and voltage variations, respectively.

## VI. REFERENCES

**[1]** G.E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," ACM Press, pp.9–14.

**[2]** A. Maiti and P. Schaumont, "Improving the quality of physical unclonable function using configurable ring oscillators,in Field Programmable Logic and Applications – FPL 2009.2009.

**[3]** F. Amsaad, T. Hoque and M. Niamat, "Analyzing the performance of a configurable ROPUF design controlled by programmable XOR gates," IEEE 58th (MWSCAS), 2015.

**[4]** C. Herder, M. D. Yu, F. Koushanfar and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," in Proc. of IEEE, vol. 102, no. 8, pp. 1126-1141, Aug. 2014A.

**[5]** A. Maiti, J. Casarona, and P. Schaumont,"A large scale Characterization of RO-PUF," Proc. HOST, 2010.

**[6]** Roel Maes, "Physically Unclonable Functions: Constructions, Properties and Applications," PhD. thesis, University of KU Leuven, Belgium, Aug. 2012.

**[7]** X. Xin, J. P. Kaps and K. Gaj, "A Configurable Ring-Oscillator-Based PUF for Xilinx FPGAs," Digital System Design (DSD), 14th Euromicro Conference on, 2011.

**[8]** Ingrid Verbauwhede and Roel Maes, "Towards Hardware-Intrinsic Security, chapter Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions", Springer, 2010.

**[9]** P.Tuyls, G.-J. Schrijen, B. Škoric, J.van Geloven, N. Verhaegh, and R. Wolters, "Read-Proof Hardware from Protective Coatings,". *In CHES 2006.*

**[10]** R. S. Pappu, "Physical One-Way Functions," Ph.D. thesis, MIT, 2001.

**[11]** P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nmFPGAs and beyond," In IEEE FPT Proc. 2006.

**[12]** J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in Symposium on VLSI Circuits, Jun 2004, pp. 176 – 179, digest of Technical Papers.

**[13]** M. Mustapa, M. Niamat, M. Alam, and T. Killian," Frequency uniqueness in ring oscillator Physical Unclonable Functions on FPGAs," in the 56th MWSCA IEEE Proceeding .Aug 2013.

**[14]** M. Mustapa, M. Niamat, ," Relationship Between Number of Stages in ROPUF and CRP Generation on FPGA," *in International Conference on Security and Management (SAM)*, Las Vegas, 2014.

# An Automated Web Tool for Hardware Trojan Classification

Nicholas Houghton, Samer Moein, Fayez Gebali, T. Aaron Gulliver
Department of Electrical and Computer Engineering
University of Victoria
P.O. Box 1700 STN CSC
Victoria, B.C. V8W 2Y2
Email: {nhoughto, samerm, fayez, agullive}@uvic.ca

***Abstract*—The complexity of modern integrated circuits has made them vulnerable to malicious insertions and modifications known as hardware trojans. A comprehensive trojan taxonomy was recently developed and incorporated into a technique for quantitatively classifying trojans. This can provide valuable insights into the origins and risk of a trojan based on the corresponding attributes, but the method can be difficult to use and prone to error. Thus, a tool has been developed to automate this classification. Widespread use of this tool will lead to a comprehensive database with detailed descriptions of all known trojans which can be used as a centralized reference for attackers and defenders. A discussion of the automated tool design is given including a case study to demonstrate its usefulness.**

## I. Introduction

Several hardware trojan taxonomies have been proposed in the literature [1]–[4]. In [1], trojans were organized based solely on their activation mechanisms. A scheme based on the location, activation and action of a trojan was presented in [2], [3]. However, these approaches do not consider the manufacturing process. A taxonomy was proposed in [4] which employs five categories: insertion, abstraction, activation, effect and location. While this is more extensive than previous methods, it fails to account for the physical characteristics of the trojan. A comprehensive taxonomy was proposed in [5] which considers the manufacturing process in characterizing a hardware trojan.

The creation of an Integrated Circuit (IC) involves numerous phases known as the IC life-cycle. As shown in Fig. 1, these phases are **specification**, **design**, **fabrication and production**, and **testing and deployment** [5]–[9]. A malicious circuit can be inserted during any of these phases, or the IC circuitry can be modified. The phase where this is done determines some of the trojan attributes. The taxonomy in [5] describes the relationship between these attributes and the IC life-cycle. It was shown that the attributes of a trojan can be used to determine in which phases a trojan can be inserted, or conversely what trojan attributes are possible if it is inserted in a given phase. However, this requires an analysis of the trojan attributes which can be tedious and thus prone to error. To alleviate this problem, an on-line tool called the Hardware Trojan System (HTS) has been developed to automate this process. With this tool, users can quickly and easily select attributes for trojan analysis. The HTS automates the necessary computations, provides documentation, and compiles a database of known trojans.

The contributions of this paper are as follows.

1) A method of analyzing trojans based on a taxonomy is proposed.
2) A web-based tool which automates the analysis of hardware trojans is described.
3) A database to store hardware trojans input to the system is developed.

The remainder of this paper is organized as follows. Section II describes the hardware trojan taxonomy. Section III introduces the classification method and its software implementation, and the HTS is described in Section IV. Section V provides a case study to demonstrate use of the HTS. Finally, Section VI presents some concluding remarks.

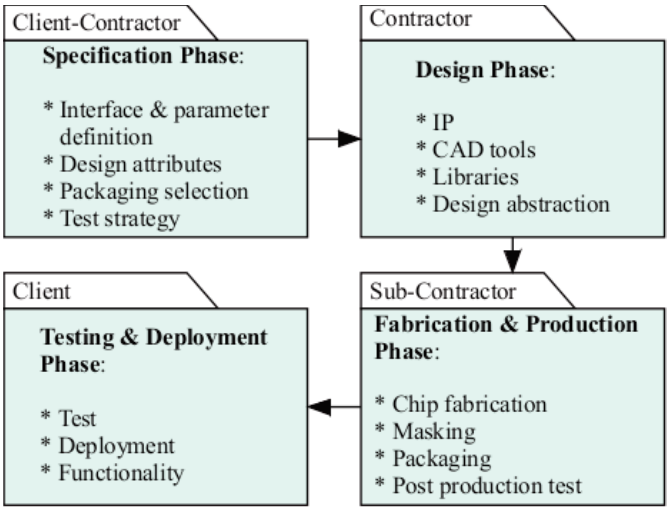



Fig. 1. The integrated circuit life-cycle [5].

## II. Hardware Trojan Taxonomy

The taxonomy proposed in [5] is comprised of thirty-three attributes. This taxonomy is comprised of thirty-three attributes organized into eight categories as shown in Fig. 2. These categories can be arranged into four levels as in Fig. 3 and described below.

ISBN: 1-60132-433-2, CSREA Press ©

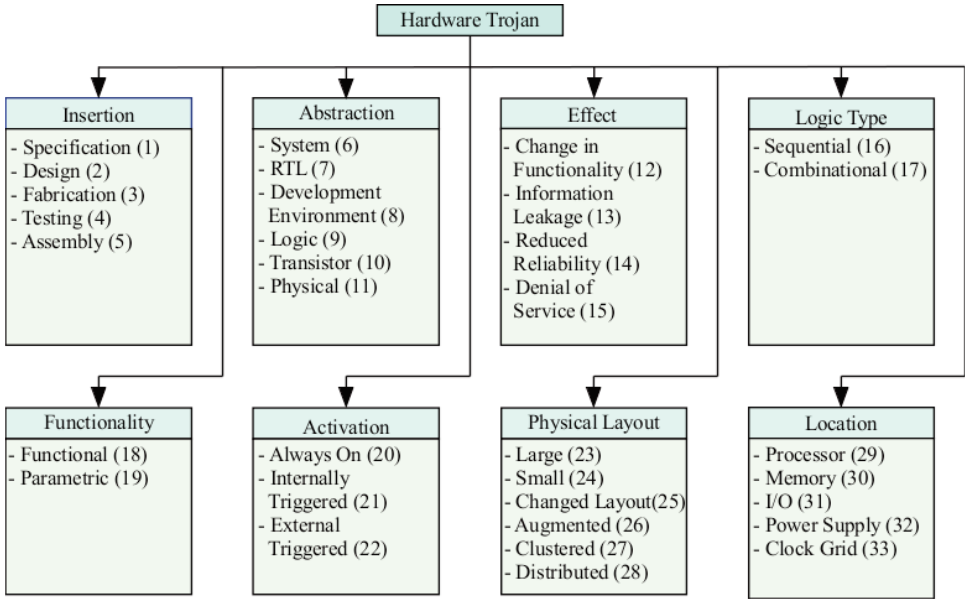



Fig. 2. The hardware trojan attribute taxonomy [5].

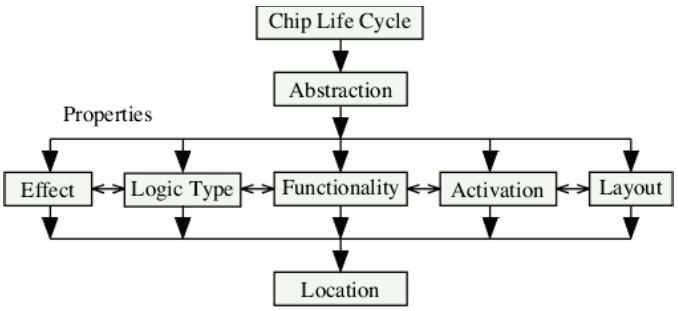



Fig. 3. The hardware trojan taxonomy levels [5].

1) The **insertion** (chip life-cycle) level comprises the attributes pertaining to the IC production stages.
2) The **abstraction** category/level corresponds to where in the IC abstraction the trojan is introduced.
3) The **properties** level comprises the behavior and physical characteristics of the trojan.
4) The **location** category/level corresponds to the location of the trojan in the IC.

The properties level consists of the following categories.

- The **effect** describes the disruption or effect a trojan has on the system.
- The **logic type** is the circuit logic that triggers the trojan, either combinational logic or sequential.
- The **functionality** differentiates between trojans which are functional or parametric.
- The **activation** differentiates between trojans which are always on or triggered.
- The **layout** is based on the physical characteristics of the trojan.

The relationships between the trojan attributes shown in Fig. 2 can described using a matrix $\mathbf{R}$ [5]. Entry $r(i,j)$ in $\mathbf{R}$ indicates whether or not attribute $i$ can lead to attribute $j$. For example, $r(2,3) = 1$ indicates that design (attribute 2) can lead to fabrication (attribute 3). This implies that if an IC can be compromised during the design phase (attribute 2), it may influence the fabrication phase (attribute 3).

The matrix $\mathbf{R}$ is divided into sub matrices as follows

$$\mathbf{R} = \begin{bmatrix} \mathbf{R_1} & \mathbf{R_{12}} & 0 & 0 \\ 0 & \mathbf{R_2} & \mathbf{R_{23}} & 0 \\ 0 & 0 & \mathbf{R_3} & \mathbf{R_{34}} \\ 0 & 0 & 0 & \mathbf{R_4} \end{bmatrix}$$

where $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$ and $\mathbf{R_4}$ indicate the attribute relationships within a category. For example, $\mathbf{R_1}$ is given by

$$\mathbf{R_1} = \left[\begin{array}{c|ccccc} A & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Submatrix $\mathbf{R_{12}}$ relates the attributes of the insertion category to the attributes of the abstraction category. An example of this submatrix is

$$\mathbf{R_{12}} = \left[\begin{array}{c|cccccc} A & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 1 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

When a trojan is discovered by a defender it can be assessed to obtain the properties according to Figs. 2 and 3. These

$\mathbf{R} =$

| $A$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| 4 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | |
| 7 | | | | | | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | | | | |
| 8 | | | | | | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | |
| 9 | | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | |
| 10 | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | | | | | |
| 11 | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | |
| 12 | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 13 | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 14 | | | | | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 15 | | | | | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 16 | | | | | | | | | | | | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 17 | | | | | | | | | | | | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 18 | | | | | | | | | | | | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 19 | | | | | | | | | | | | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 20 | | | | | | | | | | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 21 | | | | | | | | | | | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 22 | | | | | | | | | | | | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 23 | | | | | | | | | | | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 24 | | | | | | | | | | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 25 | | | | | | | | | | | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 26 | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 27 | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 28 | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 29 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 33 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

properties determine the physical location, abstraction level and insertion point based on $\mathbf{R}$. An attacker can perform a similar analysis. A detailed description of the procedures for an attacker or defender is provided in [10].

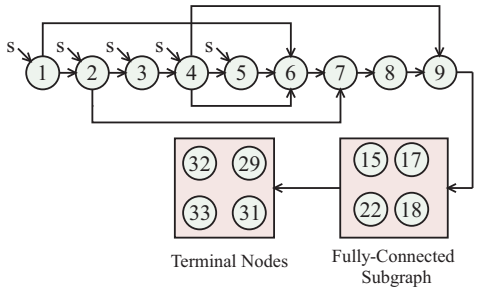



Fig. 4. The directed graph corresponding to a trojan.

## III. Hardware Trojan Classification

The classification of a hardware trojan based on the attributes in the previous section can be used to visualize the trojan using a directed graph. The attributes are denoted by nodes and the relationships between them are represented by edges. For example, consider a trojan which has attributes 15, 17, 18 and 22 and is represented by the directed graph in Fig. 4. Submatrices $\mathbf{R_1}$ and $\mathbf{R_{12}}$ indicate that attribute 1 is connected to attributes 2 and 6. This relationship can be observed in the figure. Fig. 1 indicates that the effects of a trojan in a life-cycle phase can affect subsequent phases, and this is reflected by the possible insertion points denoted by $S$ in Fig. 4. This representation of a trojan provides insight into the relationships between attributes. For example, consider an attacker who uses this method to generate the graph in Fig. 4. If they determine that it is not possible to insert the trojan in the design phase (attribute 2), then attribute 2 is removed from Fig. 4. Without access to the design phase (attribute 2), the trojan can only be inserted in the specification phase (attribute 1). Without the connection from attribute 1, attributes 3, 4 and 5 must also be removed. The trojan must access the abstraction level attributes directly via the system attribute (attribute 6).

### A. The Classification Tool

The HTS is an easy to use tool which automates the hardware trojan classification process. To investigate a trojan, users first select attributes from the categories in Fig. 2 via an easy to use selection User-Interface (UI). Once the attributes are chosen, the tool performs the necessary analysis using $\mathbf{R}$ and displays the resulting directed graph.

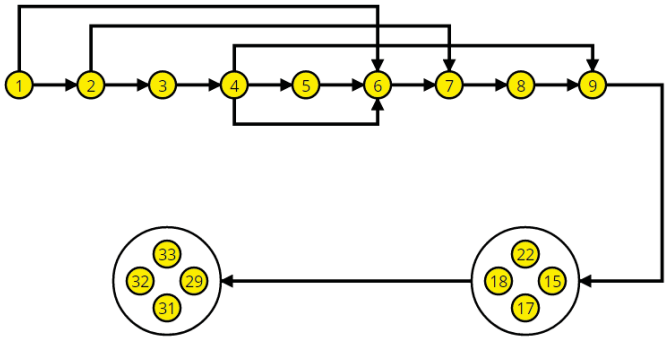



Fig. 5. The directed graph obtained by analyzing a hardware trojan.

Suppose an attacker decides to insert a trojan which has attributes 15, 17, 18 and 22. They can use the tool to obtain the directed graph shown in Fig. 5. If the design (attribute

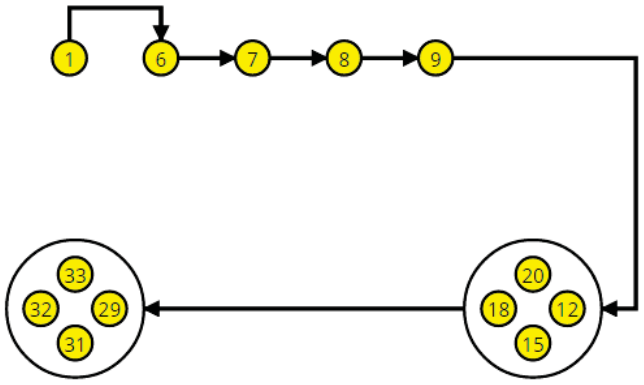



Fig. 6. The directed graph after attribute 2 is removed.

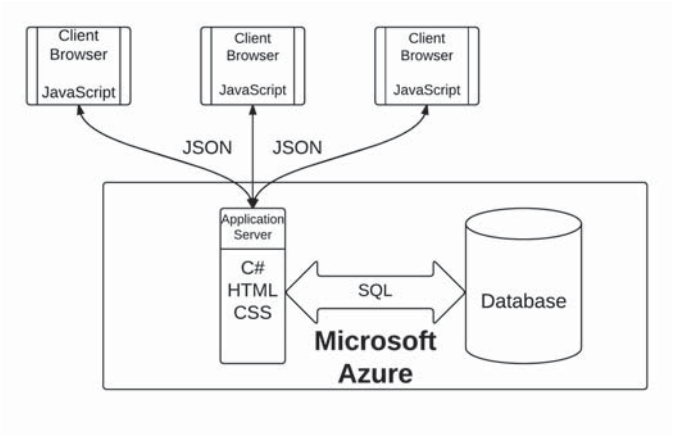



Fig. 7. Block diagram of the hardware trojan automated tool.

2) phase takes place in a secure location, the attacker will conclude that it is not possible to insert the trojan in this phase. To determine the possible trojans that can be inserted without access to the design phase, attribute 2 should be removed from Fig. 5. The classification tool provides an attribute removal feature. When an attribute is removed, the directed graph is recreated according to $\mathbf{R}$ and the new result displayed to the user. The result of removing attribute 2 from Fig. 5 is shown in Fig. 6. From the new graph, it is clear that the attacker can still compromise the system down to logic (attribute 9) in the abstraction category. The possible trojans locations remain the same, but the potential effects of the trojan have changed. Without access to the design phase (attribute 2) the trojan cannot be composed of combinational logic (attribute 17) or be externally triggered (attribute 22). Note that the attacker did not intend for the trojan to possess the attributes change in functionality (attribute 12) and always on (attribute 20), but the tool has determined that these attributes are possible.

## IV. The Web Environment

The hardware trojan classification tool was designed as a web utility for portability and easy distribution. The application server performs all of the computations and generates page markup to minimize the burden on client-side browsers. It communicates directly with a remote database used to store user account information and application data (attributes, categories and matrices). Both the application server and the database are hosted on the Microsoft Azure Cloud platform [14]. The cloud improves reliability, portability, and flexibility, provides 'on-demand' resources that are automatically managed for scalability requirements, and provides the ability for maintenance to take place anywhere. Fig. 7 gives a block diagram of the HTS. The technologies employed are as follows.

- **ASP.NET Web Form:** A user interface focused, event-driven model of the .NET framework. It allows powerful data-binding, separation of server-client side activities, a native security structure, and increased client performance [11].
- **Entity Framework:** An object-relational database mapper designed for the .NET framework. It provides a library of high speed SQL statements wrapped in C# commands to simplify development and ensure performance [12].
- **D3.js:** A Java-script library for visualizing data with HTML, SVG and CSS [13].
- **Azure:** The Microsoft cloud system [14].

Fig. 8 provides an overview of the structure of the trojan system website. The *home, contact*, *about*, and *application information* pages are accessible to all traffic. The application information page contains three sub-pages providing information on each of the primary applications. Users are required to create an account and be logged in to access the remainder of the site. Email confirmation is used to verify user accounts.

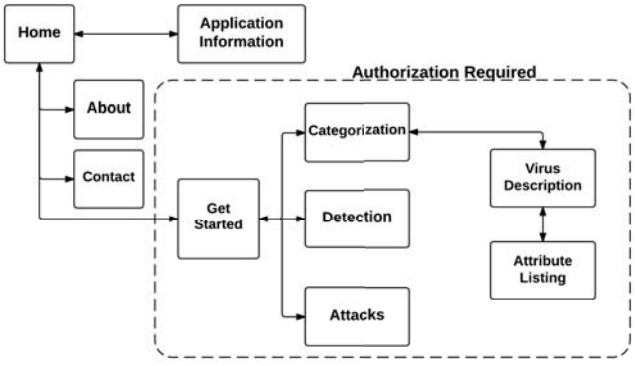



Fig. 8. An overview of the website architecture.

### A. Database

The attributes, categories, trojan descriptions and user information are all stored in a database on the *Microsoft Azure Cloud System*. The database was designed using a model first approach with the Entity Framework. The matrix $\mathbf{R}$ discussed in Section III is stored in a relational table. Each element of the matrix is stored as an entry in the table. Each table entry

contains the row number, column number and value from the matrix. The application server uses this table to perform the trojan analysis. This approach allows for the use of the highly streamlined and efficient querying power of SQL. In addition, any modifications to the matrix values or attribute definitions can be done centrally with simple Create, Read, Update and Delete (CRUD) operations. All user results are stored in the database for easy data mining capabilities. This will allow for search and statistical operations in the future.

## V. Case Study

Consider an attacker working as a design engineer in a IC manufacturing facility who inserts the circuitry shown in Fig. 9. The original output of the circuit was along wire $C$, but the addition of the trigger and payload results in $C_{modified}$. The output $C_{modified}$ is the same as $C$ except when the inputs $A$ and $B$ are both zero ($A = B = 0$). This is an example of a combinational logic triggered trojan. If $A = B = 0$ is a relatively rare event, it is unlikely to be discovered during testing. A defender who discovers this trojan can extract the attributes: change in functionality, combinational, functional, internally triggered, small, clustered, and augmented. Thus, attributes 12, 17, 18, 21, 24, 26, and 27 are the input to the classification method.

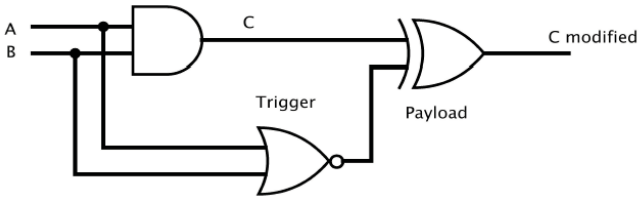



Fig. 9. A combinational trojan [10].

### A. Hand Analysis

The observed trojan attributes are all from the properties category given in submatrix $\mathbf{R_3}$. Examining the corresponding columns in $\mathbf{R_{23}}$, these attributes are directly connected to the development environment in the abstraction category (attribute 8). From $\mathbf{R_{12}}$, there is no direct connection between this attribute and an attribute in the insertion category (attributes 1 to 5). However, $\mathbf{R_2}$ provides a connection between the development environment (attribute 8) and RTL (attribute 7). Then attribute 7 is connected to design (attribute 2) and system (attribute 6). Attribute 6 is connected to specification, testing, and assembly (attributes 1, 4 and 5).

The observed properties of the trojan have now been used to analytically determine that the insertion of the trojan must have taken place during either specification (attribute 1), testing (attribute 4) or assembly (attribute 5). Further, the likely physical location of the trojan in the system can be determined. The observed property attributes (12, 17, 18, 21, 24, 26, and 27) are scanned along the rows of submatrix $\mathbf{R_3}$ and into the rows of submatrix $\mathbf{R_{34}}$. There it can be clearly seen that this trojan leads to all location attributes (columns 29, 30, 31, 32, and 33). The final directed graph is shown in Fig. 10.

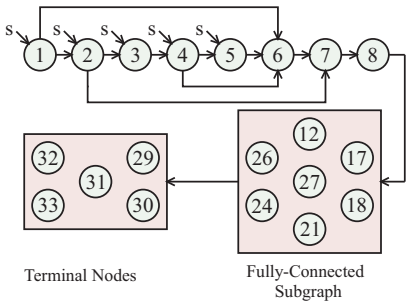



Fig. 10. The directed graph corresponding to the trojan in Fig. 9.


Fig. 11. The selection of property attributes.

### B. Classification Application

The HTS classification application avoids analysis errors by automatically producing the directed graph for the hardware trojan. Fig. 11 shows the simple UI that users employ to select attributes. Once the attributes have been selected, users are directed to the description page. The description page lists the attributes along with any relevant information as shown in Fig. 12. Users can remove attributes as required to adjust the trojan characteristics. The user can then obtain the corresponding directed graph. The analysis of the combinational trojan in Fig. 9 provides the directed graph shown in Fig. 13.

If a defender knows that a particular insertion point is secure, the tool provides a simple means of removing the corresponding attribute and recreating the matrix and directed graph. For example, suppose the attacker is aware that the testing phase (attribute 4) is not accessible. The removal of this attribute results in the revised directed graph shown in Fig. 14, which limits the insertion of the combinational trojan in Fig. 9 to the specification or design.

## VI. Conclusion

The relatively new field of hardware security requires a systematic means of investigating hardware trojans. In this paper, a comprehensive taxonomy is used in the development of an on-line suite of tools called the Hardware Trojan System (HTS). An HTS application was presented which automates

| ID | Name | Category | F_in | F_out | Select Attribute | On/Off | Remove Item |
|---|---|---|---|---|---|---|---|
| 21 | Internally Triggered | Properties | 13 | 15 | ☐ | Off | ☐ |
| 12 | Change In Functionality | Properties | 18 | 17 | ☐ | Off | ☐ |
| 27 | Clustered | Properties | 17 | 19 | ☐ | Off | ☐ |
| 24 | Small | Properties | 16 | 17 | ☐ | Off | ☐ |
| 18 | Functional | Properties | 19 | 18 | ☐ | Off | ☐ |
| 17 | Combinational | Properties | 17 | 18 | ☐ | Off | ☐ |
| 26 | Augmented | Properties | 19 | 21 | ☐ | Off | ☐ |
| | | | | | | | |

Fig. 12. The description page.

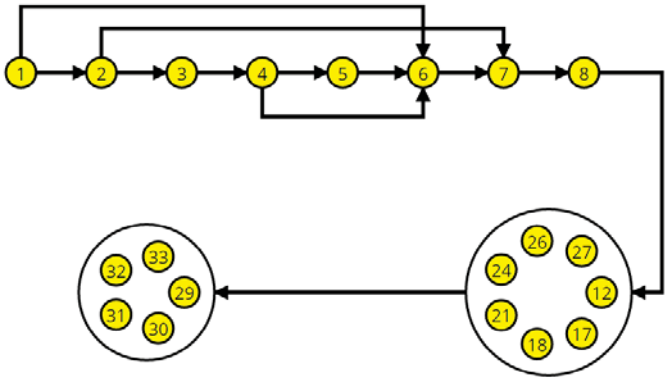



Fig. 13. The trojan directed graph obtained via the classification application.

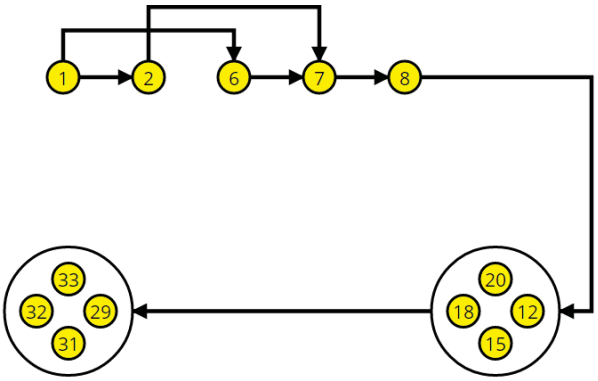



Fig. 14. Trojan directed graph after the removal of attribute 4 from Fig. 13.

the classification of trojans. This tool uses the observed attributes to automatically provide a quantitative assessment of hardware trojans. The application also compiles a trojan database.

It is left for future work to implement data mining techniques and a User Interface (UI) which allows users to browse existing trojans in the database. The addition of this functionality will provide a centralized means of evaluating the current state of hardware security.

## References

[1] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ICs: Problem analysis and detection scheme," in *Proc. Conf. on Design, Automation and Test in Europe*, Munich, Germany, Mar. 2008, pp. 1362–1365.

[2] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware trojans," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, San Jose, CA, Nov. 2008, pp. 632–639.

[3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *IEEE Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.

[4] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Proc. IEEE Int. Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, Jun. 2008, pp. 15–19.

[5] S. Moein, S. Khan, T. A. Gulliver, F. Gebali, and M. W. El-Kharashi, "An attribute based classification of hardware trojans," in *Proc. Int. Conf. on Computer Eng. and Sys.*, Cairo, Egypt, Dec. 2015, pp. 351–356.

[6] T. Zhou and T. B. Tarim, "An efficient and well-controlled IC system development flow: Design approved specification and design guided test plan," in *Proc. IEEE Int. Symp. on Circuits and Systems*, Kobe, Japan, May 2005, pp. 2775–2778.

[7] B. Sharkey. (2007) TRUST in integrated circuit program - briefing to industry. [online]. Available: http://cryptocomb.org/DARPA - Trust in Integrated Circuits Program.pdf Accessed: Nov. 11, 2015.

[8] S. Padmanabhan. (2013) Discover a better way to go from C-level to synthesis for SoC designs. [online]. Available: http://electronicdesign.com/technologies/discover-better-way-go-c-level-synthesis-soc-designs Accessed: Nov. 11, 2015.

[9] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10–25, Jan.-Feb. 2010.

[10] S. Moein, "Systematic Analysis and Methodologies for Hardware Security," *PhD Dissertation*, University of Victoria, Victoria, BC, Canada, 2015.

[11] Microsoft (2002) ASP.NET 4.5 [Computer Software] Available: http://www.asp.net/ Accessed: May 7, 2015.

[12] Microsoft (2008) Entity Framework v6.0 [Computer Software] Available: https://msdn.microsoft.com/en-ca/data/ef.aspx Accessed: May 7, 2015.

[13] Microsoft (2011) D3.js v3.5.6 [Computer Software] Available: https://d3js.org/ Accessed: May 7, 2015.

[14] Microsoft (2010) Azure Cloud System [Computer Software] Available: azure.microsoft.com Accessed: May 7, 2015.

# A Machine Learning Approach to Continuous Security Monitoring of Industrial Control Systems

Guillermo Francia, III, Jacksonville State University

Jaedeok Kim Jacksonville State University

Xavier P. Francia Jacksonville State University

**Abstract--***The need to protect our critical infrastructures, which are mostly operating through automated controlled systems, is an urgent issue that needs to be addressed. In an almost daily basis, this concern of national and international significance becomes more pronounced in light of intrusions and attempted attacks on internet-facing control systems. In this paper, we present a mechanism to apply machine learning techniques to classify, in real-time, the behavior of control systems by capturing disparate data from multiple sources and feeding those into a neural system for analysis. Through the utilization of intelligent analytics, we underscore the value of continuously aggregating and analyzing data towards the realization of an enhanced security posture of our nation's critical infrastructures.*

*Keywords:* Machine Learning, Industrial Control Systems, SCADA, Continuous Security Monitoring, System Behavior

## I. Introduction

In 2011, the National Institute of Standards and Technology (NIST) developed guidelines, published in NIST SP 800-137 [1], for information security continuous monitoring for federal information systems and organizations. That publication addresses the assessment and analysis of security control effectiveness and security posture of an organization. In early 2015, the U.S. Department of Energy's Office of Electricity Delivery and Energy Reliability published a document titled "Energy Sector Cyber Security Framework Implementation Guidance" [2] in response to NIST's Framework for Improving Critical Infrastructure Cyber Security [3]. Almost invariably, the North American Electric Reliability Corporation (NERC) continues to update and enforce a suite of Critical Infrastructure Protection (CIP) [4] standards related to the reliability of cyber security.

Clearly these guidelines and standards underscore the importance of protecting our critical infrastructures which are mostly operating through automated controlled systems. In an almost daily basis, this national need for cyber security-related CIP becomes more pronounced in light of intrusions and attempted attacks on internet-facing control systems. As documented in [5], the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) responded to 245 incidents reported by asset owners and industry partners in 2014.

Recognizing these needs, we undertook a research study on continuous security monitoring of industrial control systems using a machine learning approach. This paper presents a detailed account of the research study from inception to a partial realization of the research objectives. Furthermore, since the research study is on-going, we describe the future extensions and adaptations that we intend to pursue to enhance the security posture of control systems.

## II. The Security of Industrial Control Systems

Industrial control systems (ICS) such as those that operate our nation's critical infrastructures are beset by challenges that affect the security of their real-time or near real-time operations. Evidently, a mechanism to capture, aggregate, save and intelligently analyze disparate data from multiple sources is needed to address these security challenges. With the aid of such mechanism, operators and decision makers can take appropriate actions to mitigate harmful events, to investigate and remediate causes, and to share findings with other stake holders. However, protecting these systems can be a daunting task due to, among others the following key challenges:

i. The ability to provide a mechanism with which anomalous events are recognized in real time;
ii. The ability to collect and aggregate operational data from disparate sources such as network traffic packets of various protocols, system log files, real-time operational data from a variety of equipment such as controllers, firewalls, security appliances, tracking systems, mobile devices, human machine interfaces (HMI), database

ISBN: 1-60132-433-2, CSREA Press ©

system servers acting as historians, and remote terminal units;

iii. The ability to process these data as an aggregation of useful information in real time;

iv. The scarcity of representative samples of captured industrial control system protocol packets that can be used to train digital forensic investigators;

v. The ability to perform vulnerability assessment on industrial control systems without seriously disrupting normal operations;

vi. The difficulty of meeting regulatory compliance and standards due to ineffective data gathering and record keeping.

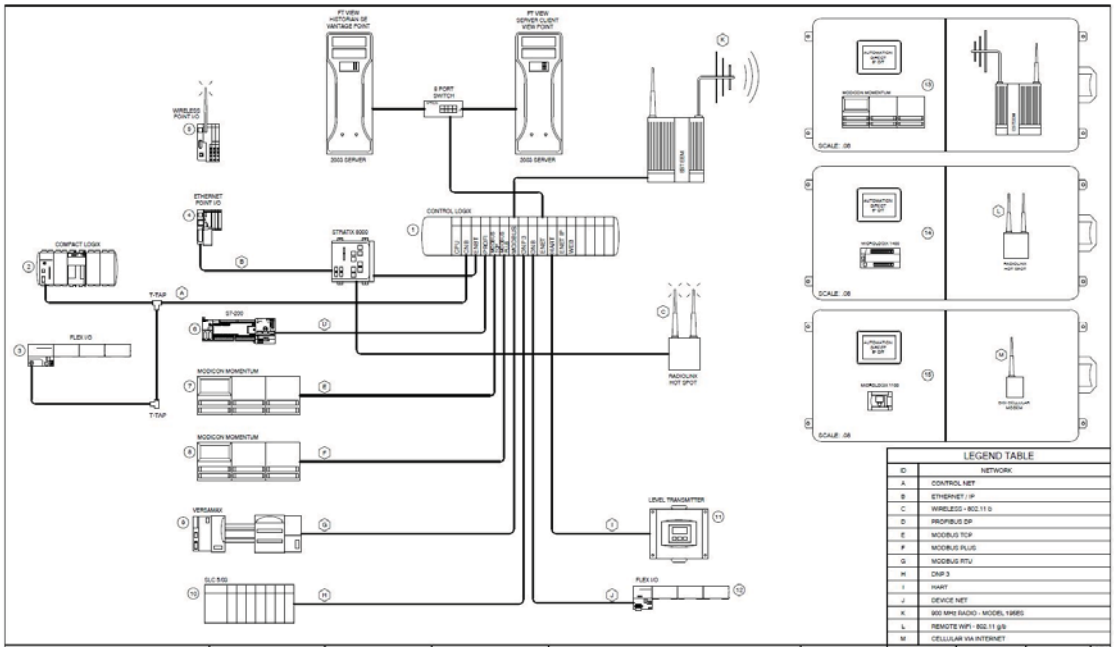
The research described in this paper is focused on finding a solution to the first challenge, i.e. finding a mechanism to recognize anomalous events in a control system environment as they occur in real-time.

## III. Facilities and Resources

In May 2010, The National Science Foundation (NSF) funded the Critical Infrastructure Security and Assessment Laboratory (CISAL) project at JSU. The system architecture of CISAL is shown in Figure 1. CISAL is designed to simulate the operations of a typical SCADA system serving a small community. The laboratory is equipped with Remote Terminal Units (RTUs), main control panels, a Human Machine Interface (HMI) and a System Historian.

As depicted in Figure 1, central to the CISAL architecture is the ControlLogix™ controller backplane. We purposely designed the laboratory with this backplane to accommodate various communication modules operating different control system protocols such as Ethernet/IP, ControlNet, DeviceNet, etc. The Remote Terminal Units (RTUs) that are integral components of the laboratory are equipped with three wireless communication modes: cellular broadband, Wi-Fi, and 900 MHz radio. Current CISAL project outcomes include the design and implementation of the control systems laboratory that mimics a water distribution system and a nuclear power generation facility, the development of various curriculum modules for a critical infrastructure protection course, and the investigation of various wireless technology applications on control systems.

Figure 1. CISAL System Architecture




## IV. Neural Network Training

The biggest challenge in implementing a neural network to conduct classification and prediction is on the training the network. Network training is the process of finding the values for the weights and biases so that, for a set of training data with known inputs and outputs, the computed outputs closely match the desired training outputs. To train a neural network, we need to measure the error between computed outputs and the target outputs of the training data. The mean squared error, which is the sum of squares of difference of two sets of outputs, is the most common measure of error. However, some of recent research results by de Boer [6] and

McCaffrey [7] suggest that using a different measure, so called Cross-Entropy (CE) error, is better performing in rare event simulations, combinatorial optimization problems and more. Consequently, the neural network algorithm involving CE error, called Cross-Entropy Method, is getting more attentions from various areas as it can be developed into a versatile tool for finding optimal solutions to many problems.

In the following discourse, we provide an overview of the process of designing a neural network model for system event classification. The linear models for classification problems are based on linear combination of fixed nonlinear basis functions $\phi_j$ and take the form

$$y(x, w) = f(\sum_{j=1}^{M} w_j \phi_j(x)), \quad (1)$$

where $w = [w_j]$ is a vector of parameters (or weights) and $f$ is a nonlinear activation function. Neural network uses this model by updating selections of parameters in each layer so that it maximizes the likelihood of generating the outputs $y(x, w)$ matching the target outputs.

The basic neural network model can be described in terms of a series of functional transformations. First, we construct $M$ linear combinations of the input variables $x_1, \dots, x_D$ in the form

$$a_j^{(1)} = \sum_{i=0}^{D} w_{ji}^{(1)} x_i, \quad (2)$$

where $j = 1, \dots, M$, and the superscript (1) indicates that the corresponding parameters are in the first layer of the network. The additional parameter $w_{j0}^{(1)}$ is the bias, and the additional input variable $x_0$ is the fixed value at $x_0 = 1$ for notational convenience. The quantities $a_j^{(1)}$ are called activations.

Next we transform each of activations using a differentiable and nonlinear activation function $h$ to yield $z_j = h(a_j)$. These quantities are called hidden units. The sigmoid function $(\frac{1}{1+e^{-a}})$ and the hyperbolic tangent function $(\tanh(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}})$ are generally chosen for the nonlinear function $h$. Now these hidden units are linearly combined to give output unit activations

$$a_k^{(2)} = \sum_{j=0}^{M} w_{kj}^{(2)} z_j , \quad (3)$$

where $k = 1, \dots, K$, and $K$ is the total number of outputs.

If we combine the two layers of neural network into a single formula, we can write it as follows:

$$y_k(x, w) = f(\sum_{j=0}^{M} w_{kj}^{(2)} h(\sum_{i=0}^{D} w_{ji}^{(1)} x_i)) \quad (4)$$

The choice of the activation function $f$ in (1.4) is determined by the type of problem. For example, if it is linear standard regression, the activation function is the identity. If we work on a probabilistic regression problem, a popular choice for the activation function is the logistic function. If it is a binary classification problem, the activation function is the nonlinear sign function: $f(x) = 1 \; if \; x > 0$ and $f(x) = 0 \; (or - 1) \; if \; x < 0$

In order to perform training algorithms, first we are given a training set comprising a set of input vectors $\{x_n\}$, where $n = 1, \dots, N$, together with a corresponding set of target vectors $\{t_n\}$. Then we can construct an error function (or a performance index) to be minimized to determine the set of parameters that may provide the maximum likelihood of target vectors $\{t_n\}$ matching network output vectors $\{y(x_n, w)\}$.

We now consider two performance indexes that are most commonly used in different algorithms. The first performance index is the mean square error (MSE) which measures the expected value of squares of difference between network output vectors and target vectors.

$$E_1(w) = \frac{1}{N} \sum_{n=1}^{N} \|y(x_n, w) - t_n\|^2 \quad (5)$$

In a single layer regression problem MSE is quadratic, the Hessian matrix [8] (or the second derivative) of the MSE function is constant. As a result, the analytic behaviors of the MSE depend heavily on the Hessian matrix. For example, if the Hessian matrix is positive definite, then the MSE has a global minimum.

The second performance index that is constructed by providing probabilistic views to the network outputs is defined as follows.

$$E_2(w) = -\sum_{n=1}^{N} \{t_n \ln y(x_n, w) + (1 - t_n) \ln(1 - y(x_n, w))\} \quad (6)$$

This error function is called cross-entropy (CE) [9]. CE measures the negative log likelihood.

## V. Neural Network Training Functions

The following step in the neural network training process is to choose a network training function (or training algorithm) that minimizes the chosen error function. Among many well-known training algorithms available in the Matlab® Neural Network Toolbox are the following: Levenberg-Marquardt [10] , BFGS Quasi-Newton [11], Resilient Backpropagation [12], Scaled Conjugate Gradient

[13] [14] and Gradient Descent with Momentum [14]. An extension of this research study is on a comparative analysis of the performances of these training algorithms on an expanded set of control systems operational data with varying data format: continuous and discrete. In this study we used the scaled conjugate gradient to train the neural network.

The conjugate gradient method uses the following steps for determining the optimal value (minimum) of a performance index $E(\mathbf{w})$. Given an initial point $(\mathbf{w_0})$ to start from, it chooses a direction $(\mathrm{p}_0)$. A line search is then performed to find the optimal distance to move along the search direction.

$$\mathbf{w}_1 = \mathbf{w}_0 + \alpha_0 p_0 \quad (7)$$

The next search direction is determined so that it is orthogonal to the difference of gradients.

$$\Delta \mathrm{g}_1^{\mathrm{T}} \mathrm{p}_1 = (\mathrm{g}_1 - \mathrm{g}_0)^{\mathrm{T}} \mathrm{p}_1 = \left(\nabla E(\mathbf{w_1}) - \nabla E(\mathbf{w_0})\right)^{\boldsymbol{T}} \mathrm{p}_1 = 0 \quad (8)$$

Repeating the two steps, we have the algorithm of conjugate gradient methods:

$$\mathbf{w}_{\mathrm{k}+1} = \mathbf{w}_{\mathrm{k}} + \alpha_k p_k \quad (9)$$

$$\Delta \mathrm{g}_{\mathrm{k}}^{\mathrm{T}} \mathrm{p}_{\mathrm{k}} = \left(\nabla E(\mathbf{w_k}) - \nabla E(\mathbf{w_{k-1}})\right)^{\boldsymbol{T}} \mathrm{p}_{\mathrm{k}} = 0 \quad (10)$$

The most common first search direction $(\mathrm{p}_0)$ is the negative of the gradient:

$$\mathrm{p}_0 = -\mathrm{g}_0 = -\nabla E(\mathbf{w_0}) \quad (11)$$

A set of vectors $\{\mathrm{p}_\mathrm{k}\}$ is called *mutually conjugate* with respect to a positive definite Hessian matrix H if

$$\mathrm{p}_{\mathrm{k}}^{\mathrm{T}} \mathrm{H} \mathrm{p}_{\mathrm{j}} = 0 \quad \text{for } \mathrm{k} \neq \mathrm{j} \quad (12)$$

It can be shown that the set of search direction vectors $\{\mathrm{p}_\mathrm{k}\}$ obtained from the (10) without use of the Hessian matrix is mutually conjugate. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction:

$$\mathrm{p}_{\mathrm{k}} = -\mathrm{g}_{\mathrm{k}} + \beta_{\mathrm{k}} \mathrm{p}_{\mathrm{k}-1} \quad (13)$$

The scalars $\{\beta_\mathrm{k}\}$ can be chosen by several different methods. The most common choices are

$$\beta_{\mathrm{k}} = \frac{\Delta \mathrm{g}_{\mathrm{k}-1}^{\mathrm{T}} \mathrm{g}_{\mathrm{k}}}{\Delta \mathrm{g}_{\mathrm{k}-1}^{\mathrm{T}} \mathrm{p}_{\mathrm{k}-1}}, \quad (14)$$

which is due to Hestenes and Stiefel [15], and

$$\beta_{\mathrm{k}} = \frac{\Delta \mathrm{g}_{\mathrm{k}-1}^{\mathrm{T}} \mathrm{g}_{\mathrm{k}}}{\Delta \mathrm{g}_{\mathrm{k}-1}^{\mathrm{T}} \mathrm{g}_{\mathrm{k}-1}}, \quad (15)$$

which is due to Fletcher and Reeves [15].

Unlike other nonlinear optimization methods that use conjugate direction (the negative of gradient), scaled conjugate gradient method does not perform a linear search when updating the vector.

## VI. Control System Data and Results

The research study involves the monitoring of a subset of a collection of complex operational data from a power generating system. We monitor the operation of three pumps, the tank water level, the operating temperature and the operational status of water-cooled equipment such as a pressured water cooled reactor or a generator. The operational and pump statuses are binary values (0-OFF, 1-ON) while the tank level is an analog value. The goal in our classification problem is to take an input vector $x$ and to assign it to a two dimensional output vector $y$ which equals either $[1,0]^T$ (normal) or $[0,1]^T$ (abnormal). Each input vector $x = [x_1, x_2, x_3, x_4, x_5, x_6]^T$ is a 5 dimensional vector where five components represent pump 1 status $(x_1)$, pump 2 status $(x_2)$, pump 3 status $(x_3)$, equipment status $(x_4$ ), operating temperature $(x_5)$ and level of water in the tank $(x_6)$. As previously indicated, the first four components $(x_1, x_2, x_3, x_4)$ hold binary information with 0 or 1 reflecting the device as on or off, respectively. Table 1 depicts the various value combinations for an abnormal condition.

We assume the following baseline conditions apply:
Tank level range: 0-99 Normal Tank Level: 36-60
High Level: >75 Critical Temperature = 325°C
Cool Down Temperature < 100°C
Critical Low Level: <10 Critical High Level: >90

Based on the perceived operational data and the baseline conditions as previously defined, we randomly generated 30,000 records of normal and abnormal system behaviors. We then partition that data into three sets: a training set, a validation set, and a test set. The training phase is allocated seventy percent (70%) of the data, which are used to determine the optimum weights on the neural network classifier. The validation phase, which tunes the parameters of the neural network to avoid overfitting, is allotted fifteen percent (15%) of the data. Finally, the testing phase, which assesses the performance (or predictive power) of the neural network classifier, is given the remaining fifteen percent (15%) of the data.

**Table 1. Abnormal Conditions**

| Condition | Pump1 | Pump2 | Pump3 | OpStatus | Temp °C | Tank Level | Status |
|---|---|---|---|---|---|---|---|
| 1 | -- | -- | 0 | 1 | -- | Any Level | Abnormal |
| 2 | -- | -- | 1 | 0 | _ | < 10.0 | Abnormal (dry tank) |
| 3 | 1 | -- | -- | -- | _ | > 75.0 | Abnormal (overflow) |
| 4 | -- | 1 | -- | -- | _ | > 75.0 | Abnormal (overflow) |
| 5 | 0 | 0 | 1 | 1 | _ | < 35.0 | Abnormal (both off) |
| 6 | _ | _ | 0 | 0 | >100 | Any Level | Abnormal (incomplete cool down) |
| 7 | 0 | 0 | 1 | 0 | >100 | < 35.0 | Abnormal (critical level + incomplete cool down) |
| 8 | _ | _ | _ | _ | >325 | Any Level | Abnormal (critical temp) |

The result of each phase of the development of the neural network classifier model is shown by their respective confusion matrix in Figure 2. To summarize, a total of 130 out of 21,000 records were misclassified in the training phase, 30 out of 4,500 records were misclassified in the validation phase; and 43 out of 4,500 records were misclassified during the testing phase. Overall, the neural network classifier system was trained to recognize normal or abnormal condition with over 99.3% accuracy. Figure 3 depicts the cross-entropy at each epoch of the development of the neural network classifier model. The cross-entropy value provides a more granular way to compute the classification error compared to the Mean Squared Error (MSE) [7].


**Figure 2. The Confusion Matrices at Each Phase of Model Development**

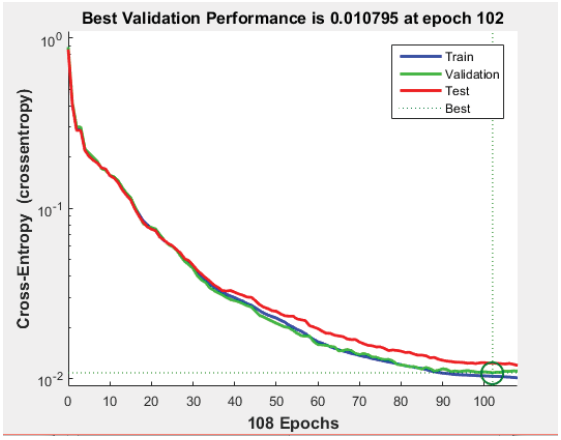



**Figure 3. The Cross-Entropy at each Epoch of Model Development**

Deployment data include 100 data items of operational data in a simulated water-cooled equipment. In order to test the predictive accuracy of our neural network classifier system model, we purposely mutated our data to include exceptions along the boundary conditions. The deployment confusion matrix shown in Figure 4 indicates the following results:

1. Out of the 100 operational data records, 25 were abnormal statuses and all (100%) of those were correctly classified.
2. There were 75 instances of normal statuses wherein one was incorrectly classified as abnormal. This yields a prediction accuracy of 98.7%
3. Overall, the neural network performed extremely well in predicting normal and abnormal system behaviors with a success rate of 99%.

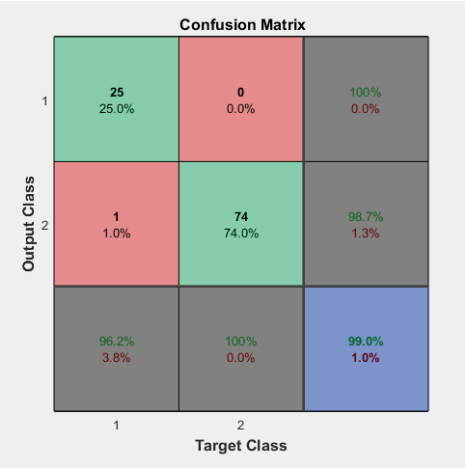



**Figure 4. The Classification of Deployment Data**

## VII. Conclusion and Future Work

The work we presented herein is part of a much larger research project on continuous security monitoring of control systems infrastructure. Industrial control systems (ICS) such as those that operate our nation's critical infrastructure are beset by challenges that affect the security of their real-time or near real-time operations. We developed a mechanism to apply machine learning techniques to classify, in real-time, the behavior of control systems by capturing disparate data from multiple sources and feeding those into a neural system for analysis. The machine learning-based data analytics scheme described in his paper highlight the value of continuously aggregating and analyzing data towards the realization of an enhanced security posture of our nation's critical infrastructures. Future plans related to this project are the following:

1. The integration of a real-time visual analytics platform;
2. A real-time alert system that depicts indicators of compromise; and
3. A threat intelligence visualization system that can be used to enhance awareness on control system security posture.

## VIII. Acknowledgement

This work is supported in part by a Center for Academic Excellence (CAE) Cyber Security Research Program grant (Grant Award Number H98230-15-1-0270) from the National Security Agency (NSA). Opinions expressed are those of the authors and not necessarily of the granting agency.

## IX. References

[1] National Institute of Standards and Technology (NIST), "Special Publication 800-137: Information Security Continuous Monitoring (ISCM) for Federal Systems and Organizations," NIST, 2011.

[2] U.S. Department of Energy, "Energy Sector Cybersecurity Framework Implementation Guidance," DOE, 2015.

[3] National Institute of Standards and Technology (NIST), "Framework for Improving Critical Infrastructure Cybersecurity," National Institute of Standards and Technology (NIST), 2014.

[4] North American Electric Reliability Corporation (NERC), "Critical Infrastructure Protection (CIP) Standards," North American Electric Reliability Corporation (NERC), 2015.

[5] ICS-CERT, "Incident Response/Vulnerability Coordination in 2014. ICS-CERT Monitor," Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), 2014.

[6] P. De Boer, D. K. Kroese, S. Mannor and R. Y. Rubinstein, "A Tutorial on the Cross-Entropy Method," *Annals of Operations Research,* vol. 134, pp. 19-67, 2005.

[7] J. McCaffrey, "Neural Network Cross Entropy Error,," *Visual Studio Magazine,* 04 11 2014.

[8] J. R. Magnus and H. Neudecker, Matrix Differential Calculus, Chichester: John Wiley & Sons, Ltd., 1999.

[9] C. M. Bishop, Patern Recognition and Machine Learning, Springer, 2007.

[10] D. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM Journal on Applied Mathematics,* vol. 11, no. 2, pp. 431-441, 1963.

[11] J. E. Dennis and R. B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Englewoods Cliffs, NJ: Prentice-Hall, 1983.

[12] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proceedings of the IEEE International Conference on Neural Networks*,

1993.

[13] M. Moller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks,* vol. 6, pp. 525-533, 1993.

[14] M. T. Hagan, H. B. Demuth and M. H. Beale, Neural Network Design, Boston, MA: PWS Publishing, 1996.

[15] L. Scales, Introduction to Non-Linear Optimization, New York: Springer-Verlag, 1985.

# Adaptable Networked Control System for Lateral Control of Aircraft Models

Kevin Terrell, Saleh Zein-Sabatto, and Mohammed Bodruzzaman
Tennessee State University, Nashville, TN. USA
(kterrell@my.tnstate.edu, mzein@tnstate.edu)

*Abstract* – **This is an exploratory research which looks into a networked adaptive control design methodology for commercial aircrafts operating under uncertain flight conditions. The methodology is developed and used to design a networked adaptive control system. A generic transport model (GTM) of commercial aircrafts is used to extract a linearized model of its latitudinal dynamics and it was utilized for testing the effectiveness of the developed adaptive control system. In this work, an adaptive controller is developed using a linear quadratic regulator (LQR) method and is used as the base of the adaptive reference control system. A reference model is designed based on the linearized model of the aircraft to simplify the design process which guarantees the stability of the developed adaptive controller. The controller is then implemented in the form of networked control system. Finally, the two different implementations of the control system are tested using the aircraft linearized model. The results obtained from testing of the directly connected adaptive controller produced an acceptable performance under different operating flight conditions. Next, a comparison between the performance of directly connected adaptive control system and the networked control system is conducted with different network configurations between the sensors, controller, and reference model. Details of the design process and the test results of the developed adaptable reference control system and the networked control system are reported in this paper.**

***Keywords: Adaptive Control; Lateral Aircraft Control; Networked Control.***

## I. INTRODUCTION

The following work is a part of the research effort of investigating the effects and performance of adaptive control system on a coupled propulsion and airframe dynamics of commercial aircrafts. The need for a better control and to maintain enhanced maneuvering capabilities of modern aircrafts specifically for a coupled dynamic has led to desire for networked control systems to better control aircrafts in general. Given a generic transport model (GTM), a linearized latitudinal aircraft model was obtained and used for this work. In this regard, several control methodologies and architectures have been reported in the literature [1, 2]. Current methodologies used to control the propulsion and airframe dynamics are not sufficient enough so that the traditional approach of designing the propulsion control system and flight control system for latitudinal control is not effective [1]. lndirect adaptive control is a control architecture that uses a controller developed from a given plant model is continuously updated using system identification techniques. A notable application of an indirect adaptive flight controller using a multiple model approach is reported in [2]. In fact, the adaptive control architecture has been investigated for quite a long time now. One of the early examples of these architectures to be flight-tested used a Receding Horizon Optimal Control Law with gains updated based on on-line parameter identification [4]. Also, there has been vast research into neural network based intelligent flight controllers [3]. A strong reason for the investigation of using a networked control system is that the network controllers allow data to be shared efficiently across the different dynamics of the aircraft [5].

This paper focuses on the development and implementation of an adaptive reference control system and a networked control methodology and is organized as follows. In Section II the technical approach used for the development of the control system is presented. Section III includes the development of the adaptive reference control system. In Section IV, the networked control system is developed and its architecture is shown and discussed.

Testing and evaluation results of the developed control system on a linearized latitudinal model of a commercial aircraft are presented as well in this section. Finally, conclusion, future work and further recommendations are provided in Section V.

## II. TECHNICAL APPROACH

The objective of this work is to further enhance the control methods of Lateral aircraft movements by developing an adaptive control system designed to assist in accurate and fast handling of aircrafts operating under different flight conditions. The research was focused on development of an adaptive based control system used for handling uncertainty that may have been injected into the model of an aircraft. In particular, this work will contribute to the development of networked based adaptive reference control methods for operating commercial aircrafts under normal and abnormal flight conditions. The full architecture of the proposed adaptive reference control system is shown in Figure 1. Generally, the architecture consists of a reference model providing desirable aircraft performance under a specific known flight condition.

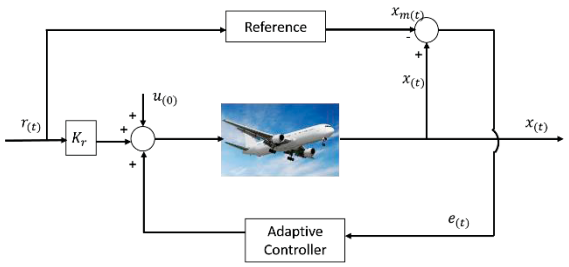



Fig. 1. Architecture of Adaptive Control System

The specific challenge we are undertaking in this work is addressed by developing an adaptive reference control system using a network strategy. The detail and specifics of the development and implementation of the adaptive control system and network strategy are presented next.

## III. DEVELOPMENT OF ADAPTIVE CONTROL SYSTEM

The design of reference model and adaptive controller are explained here. The LQR theory is used to design full state feedback controllers for the establishment of reference model. The feedback controller gain matrixes $K_{x(i)}$ were computed to minimize a quadratic cost function "*J*" in terms of the states of the aircraft and the generated control signal by the LQR controllers. The reference model will be used to determine the desired aircraft performance at the operating flight conditions of the extracted linear model of the aircraft as shown below.

$$A_{m(i)} = A_{p(i)} + B_{p(i)}K_{x(i)}, \quad B_{m(i)} = B_{p(i)}K_{r(i)}, \quad (1)$$

$$C_{m(i)} = -(A_{m(i)}x_{0(i)} + B_{m(i)}u_{0(i)}) \quad (2)$$

Where $A_{p(i)}$ and $B_{p(i)}$represent the dynamics of linear model-*i* of the aircraft at a specific flight condition, and $K_{x(i)}$ and $K_{r(i)}$ are the corresponding full state feedback controllers computed using the LQR method for the linearized models. Using the above matrices, the closed loop dynamics of each of the reference model of the aircraft becomes,

$$\dot{x}_m = A_{m(i)}\, x_m + B_{m(i)}\, r + C_{m(i)} \quad (3)$$

The design procedure proposed in reference [1] was followed to develop the adaptive controller gains used in this work. Detailed steps for the development of these adaptive controllers and proof of its stability are well documented in [1] and will not be repeated here. The equations for updating the gains of the adaptive controllers are given here.

$$\dot{K_{xi}}(t) = -S_i{}^T\, B_{mi}{}^T\, \chi_i(t)\, P_{mi}\, e_i(t)\, \Delta x^T(t) \quad (4)$$

$$\dot{K_{ri}}(t) = -\, S_i{}^T B_{mi}{}^T\, \chi_i(t)\, P_{mi}\, e_i(t)\, r^T(t) \quad (5)$$

Where $B_{mi}$ is the reference model-*i* input matrix, $\chi_i(t)$ is the indicator function given in equation (5), $P_{mi}$ is the solution of Riccati equation, $\Delta x^T(t)$ and $r^T(t)$ are the perturbation of the state and input reference command respectively. $S_i$ is set to the identity matrix. The error between the plant and the reference model is calculated using the following equation.

$$e(i) = \sum_{i=1}^{n} (X_m\,(i) - X_p(i)) \quad (6)$$

The following aircraft linearized model was extracted from the NASA Generic Transport Model (GTM-T2) and used for the development of the reference based adaptive controller. It is represented using state space matrices for an aircraft operated by four specific control components, i.e., Throttle, Aileron, Elevator, and Rudder. There are two sets of throttle engines: left and right, inner and outer engines. The aircraft has a pair of elevators, ailerons, and rudders. This research is focused on the Lateral movement of the aircraft which is controlled using the roll and yaw dynamics. The aircraft dynamics considered here is of the form.

$$\dot{x}_p = A_p x + B_p \Lambda u + B_p f + C_p; \qquad (7)$$

where the state vector is

$$x = [v, p, r, \psi]^T$$

with

$v$ = linear speed in y-direction
$p$ = body roll rate
$r$ = body yaw rate
$\psi$ = Euler angle for yaw

and the control input vector is

$$\bar{u}_p = [A, R, LSp, RSp]^T$$

with

$A$ = Aileron
$R$ = Rudder
$LSp$ = Left Spoiler
$RSp$ = Right Spoiler

The reference model of the aircraft linear mode is trimmed at yaw equal to 30 degrees with the following state values.

$x_{\text{ref}} = [3.5527e^{-26} \quad 1.5373e^{-28} \quad -1.4648e^{-28} \quad 30]$

The reference model matrix $A_{ref}$ and the control effectiveness matrices $B_{ref}$ are listed below.

$$A_{ref} = \begin{bmatrix} -0.6285 & 8.6816 & -162.5769 & 0 \\ -0.7801 & -7.9250 & 1.9022 & 0 \\ 0.2680 & -0.5153 & -1.7183 & 0 \\ 0 & 0 & 1.0014 & 0 \end{bmatrix}$$

$$B_{ref} = \begin{bmatrix} -0.0385 & 0.6034 \\ -1.4496 & 0.3583 \\ -0.0770 & -0.6595 \\ 0 & 0 \end{bmatrix}$$

The aircraft linear model used for testing is trimmed at yaw equal to 35 degrees. With the following state and input values.

$x_{01} = [-9.0291e^{-22} \quad 5.6669e^{-23} \quad -6.0632e^{-21} \quad 35]$

$u_{01} = [-0.0069 \quad 7.3156e^{-05}]$

The aircraft model matrix $A_1$ and the control effectiveness matrices $B_1$ *and* $C_{p1}$ are listed below.

$$A_1 = \begin{bmatrix} -0.6285 & 8.6816 & -162.5769 & 0 \\ -0.7801 & -7.9250 & 1.9022 & 0 \\ 0.2680 & -0.5153 & -1.7183 & 0 \\ 0 & 0 & 1.0014 & 0 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} -0.0385 & 0.6034 \\ -1.4496 & 0.3583 \\ -0.0770 & -0.6595 \\ 0 & 0 \end{bmatrix}$$

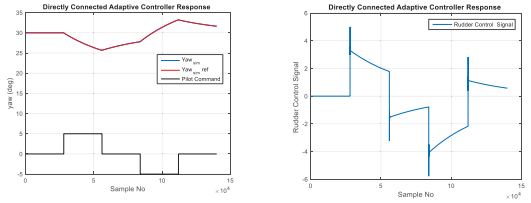
$$C_{p2} = \begin{bmatrix} -3.0908e^{-04} \\ -0.0100 \\ -4.8156e^{-04} \\ 6.0715e^{-21} \end{bmatrix}$$

The controller gains used by the reference model $K_1$ and the one used by the aircraft as adaptive controller K, are assumed equal and are shown below.

$$K_1 = \text{K} = \begin{bmatrix} 1.2838e^{-04} & 0.0022 & -0.0087 & 0.0295 \\ -0.2357 & -0.5053 & 7.9050 & 0.5316 \end{bmatrix}$$

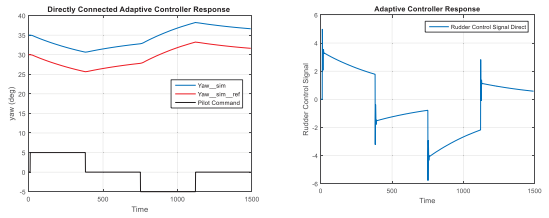
The test results of the reference model compared with the adaptive controller of the aircraft model trimmed at 35 degrees are seen in the following figures. The yaw response of first aircraft model, tested at trim values matching the reference model (30 deg) with the pilot command and the control signal, is shown in Figure 2. The response of the aircraft follows precisely the reference model with near zero error. The yaw

response of the aircraft model operated at yaw trim values of 35 degrees is shown in Figure 3. The aircraft is able to respond correctly to the pilot command and follows the reference model accurately. The above tests of the directly connected adaptive control system was implemented in a SIMULINK under the MATLAB environment.




a.) Yaw output b.) Control Signal

Fig. 2. Yaw Responses of Reference Model and Aircraft Operated at Trim yaw = 30 degrees




a.) Yaw output b.) Control Signal

Fig. 3. Yaw Responses of the Reference Model with Trim yaw = 30 deg. and Aircraft Trim yaw = 35 deg.

## IV. DEVELOPMENT OF NETWORKED CONTROL SYSTEM

The networked control system methodology is presented in this section. The communication network is developed using the SimEvents Toolbox in the MATLAB environment and used for information transfer between the aircraft output sensors, reference model, and the controller. Network control systems (NCSs) are spatially distributed systems in which the communication between sensors, actuators, and controllers occurs through a shared bandlimited digital communication network [6]. The architecture for the developed aircraft NCS is shown in Figure 4.

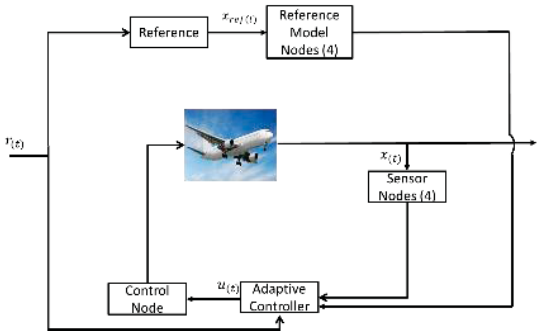
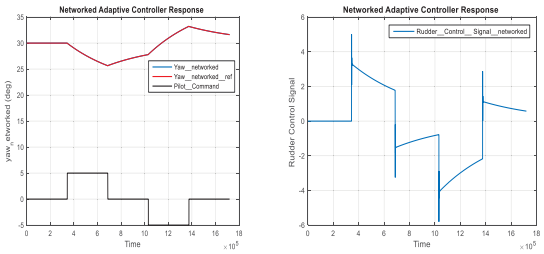



Fig. 4. Architecture for NCS

A total of nine nodes are used to establish the control system communication network. Each of the main nodes are connected as a subsystem of the network. The aircraft output sensors information are sent through a network subsystem consists of a node with a single channel and a single server for each state of the aircraft output. The communication from the reference model is done in the same manner using a single node with a single channel and a single server for each state of the reference model output. The control system output connected to a single node with two channels and one server to transmit the two control signals of the controller to the aircraft actuators. The focus of the testing is to control the rudder by sending the adaptive controller output signal to the rudder actuator using the communication network and maintain the stability of the aircraft yaw state at the same time. The implementation/testing of the NCS starts with identifying the system with a Maximum Allowable Transmission Interval (MATI) in terms of sampling time of the controller (5 msec).

The first set of testing is having a time delay induced on the control signal only. The second set of testing is with a time delay induced by the nodes from the sensors of the aircraft output while leaving all other nodes with no time delay. The final set of testing is inserting time delays on the entire network across all nodes. Figure 5 shows simulation test results of the networked adaptive control system with one sample delay allowed between the controller output and the aircraft actuator. The networked and the directly connected control systems operated similarly. In the first set of testing only the control signals' node has variable time delay induced into it and determined when the delay caused signal disturbance and complete system performance degradation. All the

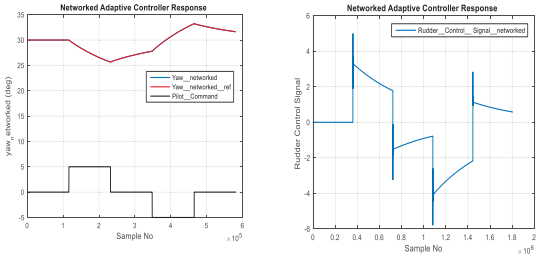
delays are tested using the trim condition of yaw = 30 for both the reference and the desired plant.




a.) Yaw State b.)Rudder Control Signal

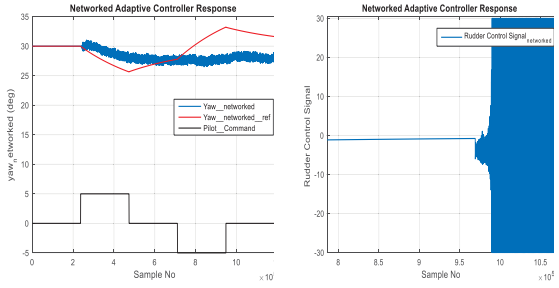
Fig. 5. Networked Adaptive Controller Response with Control Signal Delay of 1 Sample

Figures 6, 7, and 8 show right before the yaw state shows disturbance, right after the yaw signal has disturbance and lastly when the yaw signal is strongly degraded respectively.



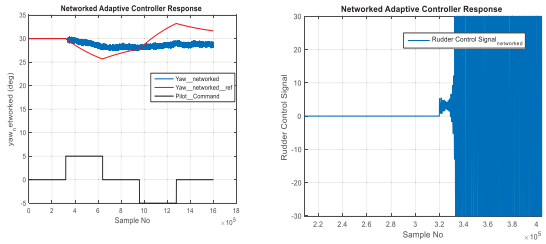

a) Yaw State b) Rudder Control Signal

Fig. 6. Networked Adaptive Controller Response with Control Signal Delay of 8 Samples




a) Yaw State b)Rudder Control Signal

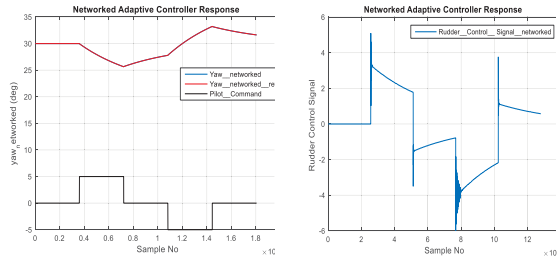
Fig. 7. Networked Adaptive Controller Response with Control Signal Delay of 9 Samples




a) Yaw State b.)Rudder Control Signal

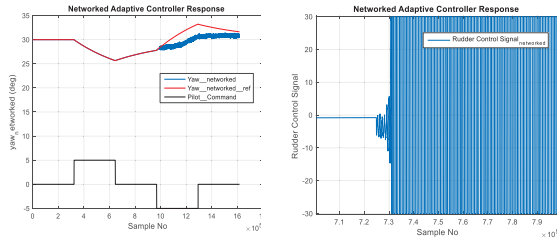
Fig. 8. Networked Adaptive Controller Response with Control Signal Delay of 12 Samples

In the second set of testing it was decided to test the system by introducing time delay on just the nodes transmitting sensor signals from the aircraft output while leaving the other nodes with no delay. The system seemed to perform roughly better in allowing a time delay of more samples before the initial signal disturbances appears. Figures 9, 10, and 11 show the results of right before any response disturbance, right after signal disturbance begins, and after total aircraft response degradation occurs respectively.



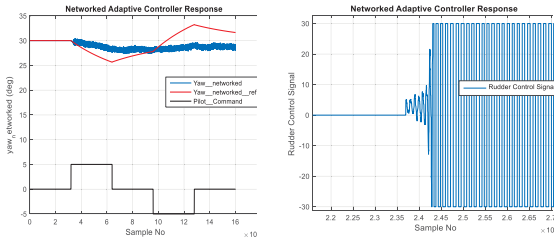

a) Yaw State b)Rudder Control Signal

Fig. 9. Networked Adaptive Controller Response with Sensor Signals Delay of 10 Samples




a) Yaw State b)Rudder Control Signal

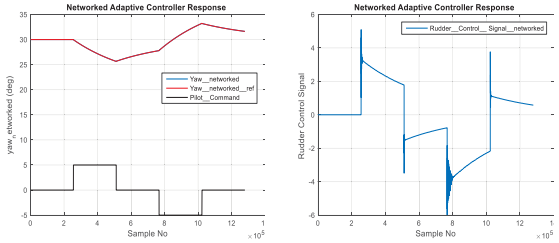
Fig. 10. Networked Adaptive Controller Response with Sensor Signal Delay of 12 Samples




a) Yaw State b)Rudder Control Signal

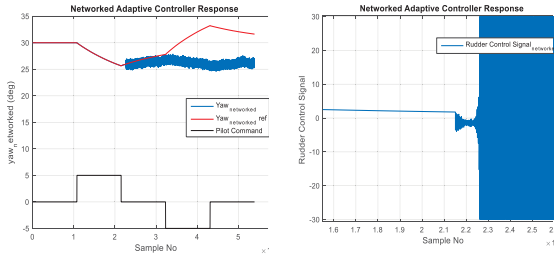
Fig. 11. Networked Adaptive Controller Response with Sensor Signals Delay of 14 Samples

For the final set of testing it was decided to test the performance of the networked control system by inducing time delay onto all nodes in the communication network as a uniform distribution delay. The system performed much worse than the other two test cases The uniform delay created less room for error on sample delay. The test results, right before any disturbance begins in the aircraft response, right after disturbance begins, and after total response degradation occurs, are shown in Figures 12, 13, and 14, respectively.



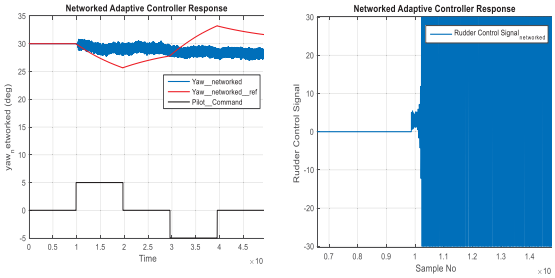

b) Yaw State b.)Rudder Control Signal

Fig. 12. Networked Adaptive Controller Response with All Signals Delay of 4 Samples




a) Yaw State b)Rudder Control Signal

Fig. 13. Networked Adaptive Controller Response with All Signals Delay of 5 Samples




a) Yaw State b)Rudder Control Signal

Fig. 14. Networked Adaptive Controller Response with All Signals Delay of 8 Samples

## V. CONCLUSION

An adaptive reference based control system was developed to control the latitudinal dynamics of an aircraft model. The directly connected (non-networked) adaptive controller achieved acceptable performance when tested under different trim conditions of the aircraft model. Next, the developed adaptive control system was implemented in the form of networked control system. The development of the communication network was focused on determining the amount of nodes needed to successfully transmit the desired data to their respective locations. As a result, networked adaptive control system was developed to connect the aircraft model with the control system. The performance of the networked control systems was tested using three node subsystems (sensors/plant, reference model and control signals). The directly connected adaptive control system performed as desired. After including the network, the system still responded well to different trim conditions with a significant time delays. The testing showed that the system performed best when there is no delay. But also the performance of the aircraft did not significantly degrade when the controller node was delayed while all other network nodes did not suffer from time delays. It has been concluded that the communication network has clear influence on the performance of the control system when it is subjected to significant time delay. However, the network can be effective and it performs its require task with a small amount of time delay. The future focus is to attempt to look into the effect of

packet loss in the network on the performance of the control system as well as the effect of queuing and gates for missing data or packets in the communication network.

## ACKOWLEDGEMENTS

The authors would to like to acknowledge NASA for providing the GTM-T2 simulation software.

## REFERENCES

[1] Sang, Q., "Model Reference Adaptive Control of Piecewise Linear Systems with Application to Aircraft Flight Control," PHD Dissertation, University of Virginia, 2012, Web. October 2014.

[2] Boskovic, D. J., and Mehra, R. K., "Multiple model adaptive flight control scheme for accommodation of actuator failures," AIAA Journal of Guidance, Control, and Dynamics, vol. 25, no. 4, pp.712-724, July-August 2002.

[3] Steinberg, M., "A Historical Overview of Research in Reconfigurable Flight Control," Proceedings of Aerospace Control and Guidance Systems Committee Meeting, Salt Lake City, UT, 2005.

[4] Bauer, Christophe, et al. "Flight control system architecture optimization for fly-by-wire airliners." *Journal of guidance, control, and dynamics* 30.4 (2007): 1023-1029.

[5] Wang, Fei-Yue, and Derong Liu. *Networked Control Systems*. London: Springer, 2008. Print.

[6] Nilsson, Johan. "Real-time control systems with delays." *Department of Automatic Control, Lund Institute of Technology* 1049 (1998).

# RAID Driver for NetBSD

Aaron Peterson and Ray Kresman

Department of Computer Science
Bowling Green State University
Bowling Green, OH 43403
{kresman}@bgsu.edu

**Abstract**

This paper describes the design of a device driver for Adaptec's RAID controller. NetBSD was chosen as the OS because of its simplicity and it lacked support for the AAC 364 raid controller. The driver is implemented in C and the paper explores the design issues in writing a driver.

## 1. Introduction

RAID is popular because of its fault tolerant capabilities. This paper describes the design of a device driver for Adaptec's RAID (Redundant Array of Inexpensive Disks) controller. The device driver was developed on a standard PC. Every PC consists of some basic components that include a motherboard, a processor, RAM, storage devices (hard drives), and a video card.

The hardware that was used in developing the device driver was a PC with enough RAM, an IDE hard drive, an AMD processor, a video card, a network card, and the AAC 364 RAID controller with 3 SCSI hard drives. The AAC 364 is a PCI card that can work on two different types of PCI buses. The card can be used on a 32bit or 64bit PCI bus. 32bit PCI buses are found in most PC's and 64bit PCI buses are found mostly in computers designed to be servers. The computer used for developing the driver has a 32bit PCI bus. The controller is plugged into the PCI bus of the motherboard. The SCSI drives are hooked up to the controller directly.

With RAID, the drivers connected to the controller are not used individually. Drives are grouped together to form what is referred to as a container. Each container can consist of many drives. So when using the computer each container will appear as one hard drive even though it consists of many drives. The controller has a BIOS that is used to set up and maintain the containers. In order for the containers to be used there needs to be a device driver. Without the device driver [1] the OS does not know how to do anything with the controller, as shown in Figure 1.

This paper describes the implementation of a device driver. With a device driver, the OS can do all of the basic operations necessary for the device to be usable. So the disk driver can do everything needed for it to make the controller and the containers usable by the OS.

Although the disk driver is fully functional it is not fully featured. There are still many improvements and features that can be added to the device driver. The first of

ISBN: 1-60132-433-2, CSREA Press ©

these features is performance enhancements. Currently, the driver only processes one request at a time. This severely limits the performance of the controller because it was built to handle up to 512 outstanding commands at one time. The device driver is written to process the 512 commands that the controller can handle but there is an error that prevents the driver from operating properly when using all 512 commands. This error is being investigated by Adaptec and myself, since it may turn out to be a problem with the firmware of the controller.

Another feature that the driver is currently missing is a management interface. This interface would allow all of the operations that are done in the BIOS of the controller, to maintain the containers, to be done by a program called the Command Line Interface (CLI). This program can do everything that can be done in the BIOS with a greater degree of flexibility and it has a larger set of features than the BIOS has. The CLI is important because without it the computer must be rebooted any time containers need to be added or maintenance needs to be done on them. This is important for servers using the controller because when the server is rebooted to do maintenance on the containers it is not available to users, so it could be difficult to find a good time to do the needed maintenance without the use of the CLI. Although this feature is something that the card can operate without it will need to be added in the future.

An interesting feature of NetBSD is that it is written to run on a wide variety of hardware platforms including PC, MAC, SPARC, etc. If a device driver is written properly it will be able to run on all of these platforms with no changes. Since the driver was developed and tested on a PC considerations for other hardware platforms were left out. Modifications can be made to the existing driver to allow it to work on the other hardware platforms but these have not been implemented in the current device driver.

**2. Overview of Design**

There are several pieces to the device driver [5, 6] that have to be written to work together to make the RAID controller functional [1-3]. All of the functionality can be obtained by writing two device drivers that consist of a total of five parts. The two device drivers are a disk driver that takes the IO requests from the operating system and the second is a driver to do all of the communication with the controller and support the management interface. Figure 2 shows the breakdown of the two device drivers.

The communication / management driver has three major parts. The first part is the probe and attach routines. These routines are used in the initial configuration of the driver to detect the device and attach the driver to the detected devices. The second part of the driver initializes and establishes communications with the controller. The third part is a management interface that is used to perform maintenance on the controller and containers.

The disk driver has two parts to it. The first part gives access to the raw devices and the second part gives access to the block devices. The disk driver's only purpose is to represent the disks on the controller to the operating system and it does no direct communication with the controller.

The raw devices are character devices that are used to do raw IO. Raw IO is done to bypass buffering built into the kernel so that maintenance commands can do reads and

writes directly to the disk with minimal kernel intervention. The raw devices are used in the initial configuration of the disk with commands such as fdisk, disklabel, newfs, fsck. More information on these commands can be seen in the man pages for each of the commands are found in [4] under "Adding a new hard disk".

The block device is the device that will be mounted. The block disk driver will take read and write requests from the operating system and the raw disk driver and submit them to the communication driver to be processed.

2.1. Probe and Attach Routines

NetBSD uses an auto configuration mechanism to detect the adapter when the computer is booting up. In order to get this to work the driver code needs to have a probe and attach routine.

When the machine boots up the device driver's probe routine is called many times with different identification information. The purpose of the probe routine is to check the identification information passed in and see if it is one of the devices that the driver supports. If it is not a supported device the probe routine returns a failure value, and if it is a supported device the probe routine returns a success value. If a success value is returned by the probe routine then the drivers attach routine will be called by the OS.

The attach sets up many different things needed to communicate with the controller. The first thing that needs to be set up to communicate with the controller are the IO registers. The IO registers are mapped into memory so that they can be accessed easily from within the device driver. The IO registers can be viewed as "doorbells" and "mailboxes". When the device driver needs to communicate a piece of information to the controller it will place the information in the mailboxes and then ring the appropriate doorbell to let the controller know that there is information waiting for it. By ringing different doorbells the information in the mailboxes has different meanings. This is the most basic form of communication with the controller.

The attach routine also establishes the interrupt service routine (ISR). Interrupts give the controller a way to notify the driver that it has information to give to it. Establishing an interrupt service routine lets the OS know what routine to call whenever it gets an interrupt from the controller.

After the IO registers are mapped and the ISR has been established the attach routine calls a routine in the communication portion of the driver to initialize the communication with the controller.

**3. Integration with the Kernel**

The ability to compile the code for the driver into the kernel is the one of the first steps taken when writing a device driver. Each kernel has a different way of loading driver code into the kernel. Some kernels support dynamically loadable modules so that only the driver module needs to be compiled and then can be loaded and unloaded without rebooting the computer. This makes the development process easier because there is no waiting for the computer to reboot every time a change is made to the driver. Unfortunately NetBSD's support for dynamically loadable modules is very limited and was not used in the development of this device driver.

NetBSD uses an Autoconfiguration method for specifying and detecting the devices that are present in the computer. Each kernel is built from a system description file. This file contains information about the computer system's hardware that the kernel will be running on. Things such as the type of processor, types of devices to support, and where those devices may be found in the computer are located in the system description file. Once a system description file is made for the computer it is then run through a program called "config". This program will copy the needed source code files to a directory and build the makefiles needed to make the kernel. Then all that needs to be done is type "make" followed by "make install" to build and install the new kernel.

In order for the "config" program to know which source files are associated with which device several configuration files are kept and "config" uses these files in setting up the kernel. To integrate the device driver into the kernel these files are modified. For a detailed description of the steps taken to integrate this device driver into the kernel see appendix A.

The communication portion of the driver initializes communication with the controller, initializes data structures used by the driver, detects the containers that are on the controller, builds and queues commands to do read and writes, and processes the completion of read and write commands. For brevity, these details are omitted.

### 3.1. The Management Interface

The management interface is used by a command line interface (CLI) to do maintenance on the containers of the raid controller. The CLI passes commands to the driver through an IOCTL interface. These commands are then sent to the communications portion of the driver to be processed. When the command has been processed the communication portion of the driver sends the result back to the management interface and the management interface sends the result back to the CLI. This part of the driver is not implemented because all management can also be done through the BIOS of the card.

### 3.2. Testing

The device driver was tested at each stage of development, but the major portion of testing was done once the driver was functional. Once the disks could be mounted and reads and writes could be processed several IO tests were performed on the driver. The simplest of the tests is to copy a large file on to the disk and then compare it to the original file to make sure that the data was copied without any corruption. This test was done using the standard Unix commands, "cp" and "diff".

Another test that was done is to use a program called "Bonnie". This program is a freely available IO benchmarking program for Unix. It was mainly used to generate IO on the disks and not for its performance measurements. Since the driver only processes one command at a time its performance is limited and its performance numbers are currently comparable to that of an IDE drive.

## 4. Concluding Remarks

This paper explored the design aspects of writing a RAID driver for NetBSD. The disk driver is fully functional, but not fully featured. For example, the driver only processes one request at a time. This severely limits the performance of the controller

because it was built to handle up to 512 outstanding commands at one time. The device driver is written to process the 512 commands that the controller can handle but there is an error that prevents the driver from operating properly when using all 512 commands. This may turn out to be a problem with the firmware of the controller.

**Reference**

1. Janet I. Egan, Thomas J. Teixeira, "Writing a Unix Device Driver", John Wiley & Sons Inc., 1992
2. NetBSD homepage. http://www.netbsd.org
3. NetBSD Documentation: Kernel. http://www.netbsd.org/Documentation/kernel/
4. The NetBSD operating system http://web.mclink.it/MG2508/nbsdeng/netbsd.html
5. The NetBSD Project. https://wiki.netbsd.org/archives/2015/01/
6. The Antti Kantee. PCI driver support for rump kernels on Xen. https://blog.netbsd.org/tnf/entry/pci_driver_support_for_rump. 9/8/2013.

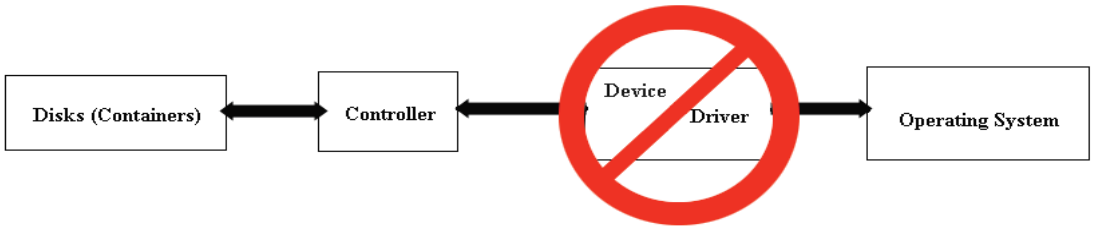



**Figure 1: Communication Path**

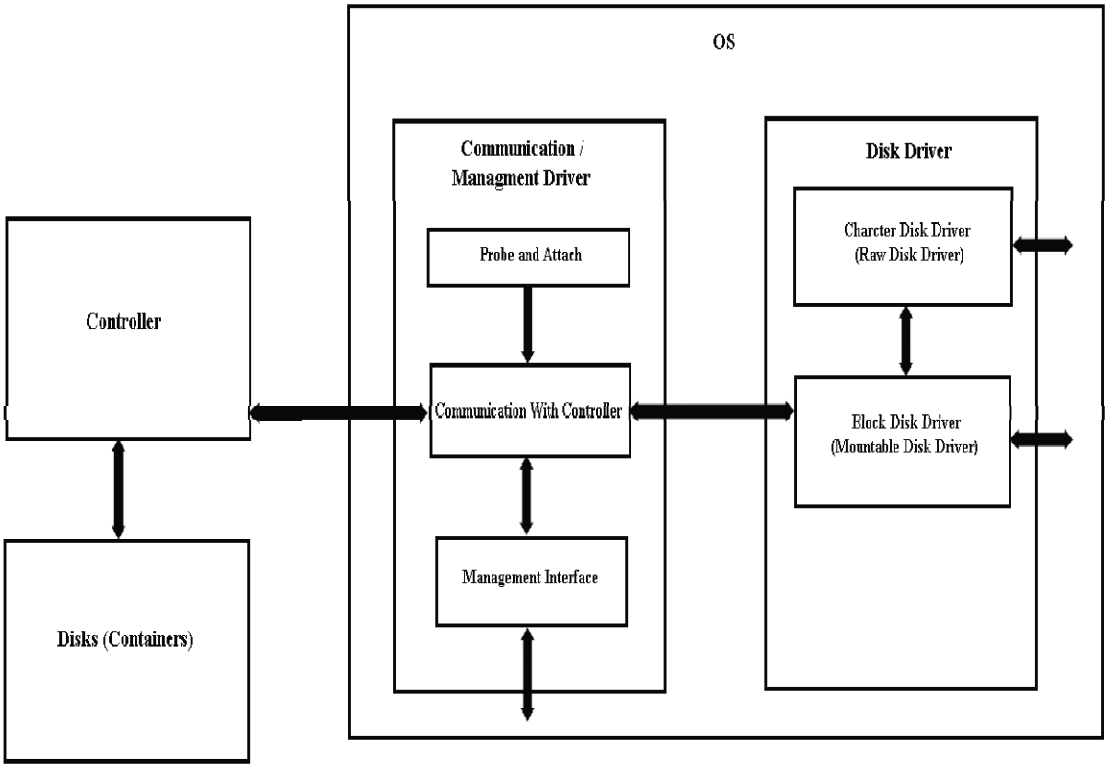



**Figure 2: Driver Block Level Schematic**

# Analysis of ICS and Corporate System Integration Vulnerabilities

**K. ES-SALHI**[1], **N. Cuppens-Boulahia**[1], **D. Espes**[2] **and F. Cuppens**[1]
[1]LabSTICC, Telecom Bretagne, Cesson sevigne, France
[2]LabSTICC, University of Western Brittany, Brest, France

**Abstract**— *Integrating industrial control systems (ICS) with Corporate system (IT) is becoming a necessity in the current industrial business world. The benefits of this integration are numerous but not without a strong risk. Indeed, the integration of ICS and IT systems exposes both to many security problems. Hence, securing this interconnection becomes one of the major priorities of the industrial sector. In this paper, we present a summarized overview of integrated ICS vulnerabilities, we evaluate the state of art of existing countermeasures and we suggest some research subjects that may be necessary to ensure integrated ICS security. For this purpose, we have studied several known security standards such as ISA, NIST and other papers on ICS systems convergence.*

**Keywords:** Cyber security, Vulnerabilities, ICS, SCADA, ICS Corporate systems Integration.

## 1. Introduction

*Industrial control system* (ICS) is a general term that encompasses several types of control systems used in industrial production, including *supervisory control and data acquisition* (SCADA) *systems*, *distributed control systems* (DCS), and other control system configurations often found in the industrial sectors and critical infrastructure [1].

The industrial business world presently has a real need to integrate industrial control systems (ICS) with Corporate systems [2]. The benefits of this integration include increased visibility of industrial control system activities, ability to use business analysis to optimize production processes and gaining more responsiveness to the business requirements and decisions. For example, ERP [1]and CRM systems [2]

Interconnecting the two networks introduces multiple security problems. Indeed, industrial systems have been designed without security in mind because they have usually been isolated. This is reflected by the use of insecure industrial equipment (devices, protocols ...), the lack of clear security policies and the lack of human resources security training. Therefore, integration exposes both ICS and Corporate systems to major security threats. Moreover, ICS and Corporate systems are different by their nature and have different security requirements in terms of confidentiality, integrity and availability. ICS security focuses on availability, plant protection, plant operation, control complexity and time-critical systems response while Corporate security focuses on protecting information. Hence, responding to the integrated system security's needs cannot be accomplished using only known IT security skills. In addition, ICS teams have unfortunately no knowledge about cyber security and, on the other side, IT security specialists have insufficient knowledge about industrial systems.

This paper presents an overview of Integrated ICS systems vulnerabilities and countermeasures identified by other studies and security standards trying to answer the following questions:

- *What are the identified problems and needs?*
- *What security measures they suggest?*
- *Are the existing Integrated ICS security countermeasures sufficient and implementable?*

We will especially focus on vulnerabilities and countermeasures directly related to the integration between ICS and Corporate network. Internal ICS network vulnerabilities will not be covered by this paper.

Section 2 presents architectural models of integrated ICS systems in order to illustrate ICS and Corporate systems boundary. Defining this boundary was very important to classify integration vulnerabilities. These vulnerabilities will be presented in Section 3. Section 4 discusses the existing countermeasures. It provides an analysis of the sufficiency of these measures as well as some suggestions for future work.

For the rest of this document, we will use the abbreviation IICS to refer to "Integrated ICS". An IICS is an ICS that is integrated with its organization's Corporate system.

[1]Enterprise Resource Planning is a category of business-management software i.e., typically a suite of integrated applications that an organization can use to collect, store, manage and interpret data from many business activities.

[2]The Customer Relationship Management is a type of business software for capturing, processing and analyzing information on customers and prospects in order to retain them by providing optimized services.

## 2. IICS architecture

IICS systems have various architectures and infrastructure compositions depending on different aspects such as the industrial sector, the production activities and the size of the organization. Actually, industrial sector determines the needs within the industrial system and has a direct impact on the infrastructure configuration and architecture of the IICS.

There are two main industry sectors :

- Manufactoring industries including "Continuous manufactoring Processes" (e.g., fuel or steam flow in a power plant) and "Discrete-based manufactoring industries" (e.g., Electronic and mechanical parts assembly),
- and Distribution industries (e.g., water distribution and wastewater collection systems).

The two industry families have different characteristics and needs. For example, in terms of localization, manufacturing industries are usually located within a confined factory or plant-centric area, whereas distribution industries are geographically dispersed. Moreover, communication in manufactoring industry are usually performed using LAN while distribution industry systems usually communicate through WAN [1] [3].

Identifying vulnerabilities of all IICS configurations is unfeasible. Therefore, the goal of the first part of our work was to establish a reference architecture. This will be presented in the next subsection. Besides, ICS and Corporate system integration requires enclosing the boundary between them. This boundary is identified using relevant models that represent functions, physical equipment, information within the whole system.

### 2.1 Reference architecture

Our study is mainly based on the hierarchical functional model provided by ISA-95 [4]. This model is based on the Purdue Reference Model for CIM on the MESA International Functional Model. It depicts the different functional levels (figure 1) that define hierarchical levels at which decisions are made and underlines the relationship between Business and Industrial activities. In other words, it gives a high level picture of the functional architecture of an IICS. It also outlines ICS and Corporate system functional integration at the interface between Level 4 and Level 3. This interface is between plant production scheduling and operation management and plant floor coordination.

The five functional levels inside an IICS are [4]:

- **Level 4 - Enterprise Business Systems**: This level is described as "business planning and logistics". It includes the functions involved in the business-related activities.
- **Level 3 - Operations Management**: This level includes the functions involved in managing and optimizing the production work flows.
- **Level 2 - Supervisory Control**: It includes the functions involved in monitoring and controlling the physical process.
- **Level 1 - Local or Basic Control**: It includes the functions involved in collecting data and manipulating the physical processes. This level contains process monitoring, safety and protection equipment that monitor the industrial processes and automatically return them to a normal state when unwanted events occur.
- **Level 0 - Process**: It is the actual physical process. It includes sensors and actuators directly connected to the production process.


Fig. 1: Functional Hierarchical model [3].

### 2.2 IICS main components

IICS functions involved in the 5 levels listed above, are implemented using multiple software and hardware components. Figure 2 provides a quite complete list of these components relating them to the hierarchical model.

1) **Level 4: Enterprise and Business system**
   - **ERP:** Enterprise Resource Planning is a category of business-management software i.e., typically a suite of integrated applications that an organization can use to collect, store, manage and interpret data from many business activities.
2) **Level 3: Operational Management**
   - **MES:** Manufacturing Execution System is a computerized system used in manufactoring to track and document a manufactoring process. MES can provide the right information at the right time and show the manufacturing decision maker "how the current conditions on the plant floor can be optimized to improve production output"[1] MES might be seen as an intermediate system between an Enterprise Resource Planning (ERP) system,

and a Supervisory Control and Data Acquisition (SCADA) or process control system.

3) **Level 2 : Supervisory Control**
   - **SCADA:** Supervisory control and data acquisition systems integrate data acquisition systems to provide a centralized monitoring and control systems for numerous process inputs and outputs. Supervisory control layer contains the top level components of SCADA to which we refer as control centers.
   - **Control Center:** It is the central part of a SCADA system. It collects and logs information received from other SCADA components and may generate actions based upon detected events. It is composed of the following elements:
     – **SCADA control Server** (usually called the master): It is a software service that is connected to field devices using industrial protocols, exposes acquired supervision data, and sends controls.
     – **HMI**: It is an operators interface specifically designed for monitoring and control of other SCADA components.
     – **Historian**: It is a data storage server that is used to store history data.
     – **Engineering workstations**: It is a computing unit, generally an ordinary computer, that industrial engineers mainly use to configure industrial control components (especially PLCs and RTUs...).
     – **OPC**: It is a software interface standard that allows Windows programs to communicate with industrial hardware devices.

4) **Level 1: Control**
   - **Field Sites:** Field sites are local or remote and are composed of devices directly responsible of data acquisition, logical operations and controls. A field site contains generally:
     – One or more **PLC** or **RTU** which are central sophisticated devices that directly interact with sensors and actuators in the site.
     – And sometimes a local computer connected to RTU/PLC for configuration.
     – **PLCs** are digital devices used for automation of industrial electromechanical processes. They connect to sensors in the process and convert their signals to digital data.
     – **RTUs** (remote terminal unit) are microprocessor-controlled electronic devices that interface sensors and actuators by transmitting telemetry data to the SCADA Server, and by using messages from the master supervisory system to control connected objects.

5) **Level 0: Process**
   - **Sensors and Actuators:** Devices involved in the actual physical industrial process.

## 2.3 IICS logical architecture

The previous model also highlights the functional boundary between Enterprise system and Industrial system. It is illustrated by the dashed line between level 4 and level 3. We are especially interested in the exchanges and data flows that cross this line. Our IICS architecture is composed of three zones:

- Enterprise zone: Level 4
- Enterprise, industrial system integration zone: Interface between Level 4 and Level 3
- Industrial system zone (ICS zone): This zone is the actual ICS including all the 4 lower levels of the hierarchical model.

IICS functions involved in the 5 levels listed above, are implemented using multiple software and hardware components. Figure 2 illustrates a logical reference architecture that will constitute the basis of our work. This figure also illustrates the three IICS zones.

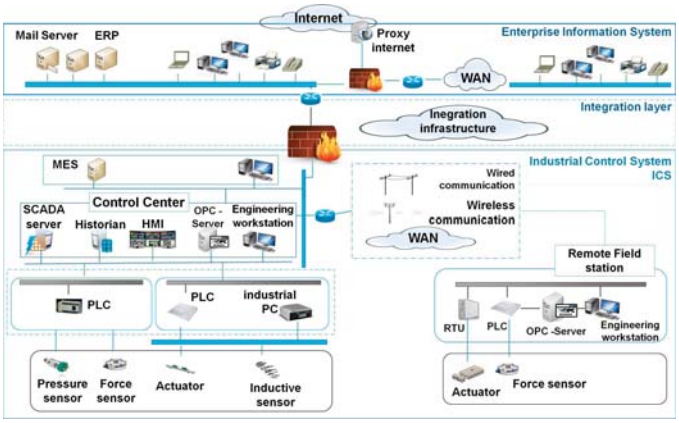



Fig. 2: Logical Reference Architecture.

Communication across the integration zone is needed for multiple purposes and involves different components. For example, coordination between business and industrial activities usually require connecting Enterprise Resource Planning (ERP) system with Manufactoring Execution System (MES) or SCADA historian. Table 1 gives some examples of flows considered to be necessary between the Industrial system and the Corporate system.

# 3. Vulnerabilities and challenges

By connecting ICS and Corporate networks, a lot of security problems arise. It is mainly because ICS networks are heavily predisposed to be vulnerable. The predisposing

Table 1: Some examples of flows between ICS and Corporate systems [5].

| Flow Type | Example | Used protocols |
|---|---|---|
| Applicative data flows | -Production scheduling data -Production Cost accounting -Industrial Processes supervision data -Material and energy management data | JCA, Web Services (Http(s), FTP(s), SQL) TCP;IP |
| Supervision data flows | Devices supervision and monitoring | PING ICMP, SSH, TELNET,TCP,IP |
| Other flows | Emailing | Http(s), SMTP(s), IMAP(s), POP |
| | DNS | |
| | Printing | TCP |
| | Remote maintenance | |

conditions range from human and organizational problems such as no training of personnel or lack of security policy, to Infrastructural and Architectural problems such as use of insecure protocols, lack of consideration of security in architecture and design. Moreover, low-cost IP-based devices are now replacing proprietary solutions, which increases the possibility of cybersecurity vulnerabilities and incidents.

IICS systems could be highly vulnerable if no countermeasures are taken. According to many research works [1], [3], [6], [7], [8], [9], a quite exhaustive list of vulnerabilities issued by IICS systems can be provided. We collected, grouped and evaluated these vulnerabilities in order to assess the state of art and identify issues that are not (or not completely) addressed yet. We are particularly interested in vulnerabilities related to ICS integration.

IICS vulnerabilities can be categorized in two categories [1]:

1) **Policy and Procedure vulnerabilities:** This category contains the vulnerabilities that arise because of incomplete, inappropriate, or nonexistent security policy, including its documentation, implementation guides (e.g., procedures), and security enforcement.

2) **System vulnerabilities:** This type of vulnerabilities includes:
   - Architecture and Design vulnerabilities,
   - Communication and Network vulnerabilities,
   - Configuration and Maintenance vulnerabilities,
   - Physical vulnerabilities,
   - Software Development vulnerabilities.

Configuration and Maintenance, Physical and Software Development vulnerabilities already exist in both ICS and Corporate systems even without them being integrated. They are less impacted by the integration process. However, Architecture and Design vulnerabilities, Communication and Network vulnerabilities and Policy and Procedure vulnerabilities can be severely strengthened by the interconnection of the two worlds as they are directly related to the integration process. We particularly focus on these types of vulnerabilities.

For the sake of this study, we made a quite exhaustive list of IICS vulnerabilities that belong to these categories. Below a summary of these vulnerabilities:

1) **Architecture and Design vulnerabilities**

This category is tightly related to ICS integration process. In fact, integrating an ICS system with a Corporate system consists of interconnecting components from the ICS system to components from the Corporate system for some functional or technical purpose ensuring their communication, their interoperability, their security as well as the whole system security. Unfortunately, ICS architectures are definitely not designed to be integrated with Corporate systems architectures from a security perspective. Indeed, most of ICS architectures do not make use of segmentation which means that all the components in the system are on the same low level of security. Besides, authentication, encryption of exchanged and stored data, logging and traceability mechanisms are mostly absent and the majority of adopted technologies represent serious sources of vulnerabilities. For example, some devices are implemented with no security capabilities (authentication and encryption ...) at all. Some others come with hard coded passwords on their firmware and cannot thus be efficiently secured.

In addition, industrial temporal requirements prevent equipping components with anti-virus and firewalls to optimize resource usage. Moreover, applications and drivers inter-compatibility requires keeping them on very precise not up to date versions. Even for the most carefully designed ICS systems, cyber security is not seriously applied. For example, proprietary protocols are thought to be secure without any verification and architectural choices tend to very quickly be taken without security consideration.

Therefore, it is really necessary to pay special attention to architectural aspects at the integration design and

build time. Designers should, in particular, take the existing heterogeneity into account and opt for an architecture that protects against vulnerabilities such as :

- Backdoors and "holes" (either intentional or not) in the network perimeter [7],
- Inadequate segregation [8],
- Usage of technologies with known vulnerabilities [10],
- ICS network used for non-control traffic [1].

Cautiousness should also be exercised when applying architecture modifications and configuration updates to an existing integration layer.

2) **Communication and Network vulnerabilities**

ICS and Corporate networks integration introduces new communication and network security needs that have never been addressed. Indeed, integration puts a lot of heterogeneous protocols together. On one hand, IT protocols heavily focus on data security and can unfortunately not be used in ICS network in which protocols focus more on availability. On the other hand, ICS protocols (e.g., Modbus/TCP, Ethernet/IP, DNP3) are not designed with security in mind and, worse, the most vulnerable among them are sometimes employed despite their known vulnerabilities. For example, DCOM, a protocol used for OPC, uses RPC which opens multiple ports to establish communication making firewalls configuration difficult. More seriously, with the recent trend of using Commercial of the shell technologies (especially open protocols such as TCP/IP protocols stack and ordinary Windows computers), ICS systems are more vulnerable than ever before because these technologies knowledge is easily available to attackers.

3) **Policy and Procedure vulnerabilities**

The current ICS systems also present a lot of human related security problems. These problems are direct consequences of two main factors:

a) The lack of personnel training and sensitization program.
b) The absence of a well-defined globally applied security policy that establishes suitable security procedures and constrains human resources to adopt convenient security practices.

As examples of these security policy problems, we may list [1]:

- Lack of configuration management policy.
- Lack of adequate access control policy.
- Lack of adequate authentication policy.
- Inadequate security incident detection and response plan.

Dealing with this type of vulnerabilities in an IICS is quite challenging. ICS and Corporate systems have always formed two separate entities managed by different teams. Corporate systems security policies are significantly more mature than ICS's ones when they exist. The main difficulty is to define a common security policy that takes into account Corporate and ICS specificity. Corporate security management teams are more qualified to define this common security policy but they do not have enough experience with ICS networks. Defining a concise efficient global IICS security policy is hence far from being straightforward.

# 4. Analysis and suggestions

For each identified vulnerability, we extracted the related countermeasures suggested by the studied documents, we compared them and evaluate their efficiency to achieve security objectives, their maturity, and their implementability.

Each extracted countermeasure belongs to one of the three following countermeasures families [3]:


Fig. 3: Countermeasures families.

- **Measures on "People":** Assuring employees awareness and training,
- **Measures on "Process":** Defining security policies and procedures as well as identifying the real security needs,
- **Measures on "Technology":** Securing equipment and technologies.

NIST-800-82 Annex G [1] (for ICS security measures) combined with NIST-SP-800-53 Annex G [11] (for Corporate system security measures) constitutes one of the most valuable resources on IICS security measures. These two annexes are essentially equivalent except that the first adds specific considerations about ICS security measures. When combined, they offer an absolutely exhaustive list of IICS security measures. The documents we studied, especially these two annexes, generally cover very well

"people" and "process" security aspects. Indeed, these annexes define multiple groups of measures for which they provide important guidance on security policies and procedures. At least, they can be used as a check list that helps to select adequate "people" and "policy" measures for the system to be secured.

Concerning "Technology" vulnerabilities, especially architecture and communication vulnerabilities, the studied documents recommend a set of countermeasures that should be applied to the integrated system. As regards architecture and design vulnerabilities, it is recommended that security be considered from the beginning of any integration project in order to respond more efficiently to both security and operation requirements. This should be done on a case by case basis during the definition phase in which the security perimeter has to be precisely defined. Special attention should be paid to the ICS system temporal and availability requirements as well as risk and safety requirements. Designing the final solution should take into account the existing systems, their components, their communication flows and their heterogeneity. This helps to figure out the upcoming complexity and adequately select measures needed to meet the defined requirements.

Nonetheless, there are still multiple generic security measures that are advocated for IICS systems. Defense-in-depth is one of the most important of them. This very common security technique is mainly based on segmentation and segregation. The latter consists of enforcing a rule-set to control permitted communications through the segments boundaries using firewalls and intrusion detection systems (IDS). Defense-in-depth should be applied at the IICS integration layer by adding a demilitarized zone (DMZ) as a separate segment that only contains the ICS component connected with Corporate system.

Furthermore, multiple other security measures such as using authentication and authorization mechanisms, using secure protocols, monitoring and logging mechanisms, single point of failure, redundancy techniques are also strongly recommended by the studied documents. For a quite exhaustive list of these measures, please refer to the following NIST standards (Annex G) [1] [11].

Despite the amount of useful information provided by the studied documents, there are still several "Technology" vulnerabilities measures that either are absent or need to be completed or detailed. Firstly, most of the suggested countermeasures are directly borrowed from the IT world and do not really take ICS specificity into account. Besides, most of these countermeasures only serve for guidance or as checklist as they lack reference implementation and real application examples. More concisely, the studied documents dictate more what to do than how to do it. That is why we argue that, currently, there are a lot of IICS security requirements that cannot yet be satisfied using the existing solutions. Among the subjects that need more work, we can list :

1) **IICS Authentication and Authorization mechanisms:**

   Authentication and Authorization mechanisms are highly recommended for IICS systems. There are mainly two types of these mechanisms: distributed authorization and authentication solutions and centralized ones. Selecting one solution depends on the organization structures and scalability requirements. However, there are many challenges to integrate authentication solutions, especially centralized solutions, into an IICS [1]. This is because ICS may utilize its own application-specific accounts and authentication mechanisms that are not designed to be integrated with third party servers and protocols. Besides, most industrial equipment do not have any authentication support and even less capability to be integrated with centralized authentication solutions such as AAA architectures.

   Using centralized authentication solutions in IICS systems and using AAA architectures and protocols for ICS systems are subjects that still need more work.

2) **IICS Segmentation and Segregation:**

   As explained before, segmentation and segregation are very important security techniques for IICS systems. Unfortunately, we could not find enough details on how to efficiently implement them in an IICS context. Indeed, there are still several questions that remain unanswered [7], [1]. First, no precise explanation was given on how to partition IICS networks into segments. Should the segmentation be based on the components'physical characteristics, on their functions or on their geographical localization? In addition, although segregation rule-sets definition is not trivial, no sufficient explanation was provided.

   Furthermore, segmenting large-scale networks is a complicated task for administrators. It is all the more complicated in systems with frequently changing configuration and topology. We did not either find enough information on how to perform segmentation in large-scale networks taking into account architecture changes and configuration updates. This is why we are convinced that IICS Segmentation and Segregation are major elements to

be studied.

3) **IICS firewalls:**

Segregation ought to employ firewalls to control communication across segment boundaries. To the best of our knowledge, there is unfortunately no firewall fully adapted for all industrial systems configurations [7], [1]. Existing firewalls only support protocols commonly used in Corporate systems especially TCP/IP and, for the best, some industrial protocols such as ModBus and DNP3 (see Tofino Industrial Security Solution [12]). IICS systems are really in need to industrial firewalls that can be used with different industrial systems configurations (distributed, located, wired, wireless...) taking into account industrial technologies and protocols and more importantly industrial timing requirements.

4) **IICS IDS/IPS:**

Similarly, IDS/IPS are really indispensable for IICS systems. Unfortunately, there is no commercial IDS/IPS solution that is really adapted to IICS systems. This is because existing IDS/IPS solutions do not support industrial protocols, and are not designed to respect ICS requirements. Besides, no precision was given on where to place IDS within an ICS system.

As far as we are concerned, we will, for the next step of our work, focus on IICS Segregation. It is true that ISA99 [3] and ICS-CERT [7] give some guidelines and recommendations on how to implement IICS Segregation. However, we are convinced that the provided information is not enough detailed and precise. Indeed, the given guidelines do not answer our questions about Segregation. Hence, we suggest to further investigate this issue taking into account the multiple aspects of IICS systems division such as geographical localization, business orientation, hierarchical and functional orientation as well as scalability requirements.

## 5. Conclusion

This paper highlights one of the major challenges of current Industrial world namely how to integrate two completely different networks (ICS and Corporate system) and keep the integrated system secured. It provides a very useful reference architecture that can be used by other works, a summarized overview of identified vulnerabilities, and an analysis of countermeasures evaluating their efficiency and suggesting some research issues to complete the existing solutions panel in order to secure IICS more efficiently.

A lot of work has been done on IICS security. Multiple vulnerabilities have been identified and a lot of countermeasures have been suggested. However, ICS and Corporate systems integration remains very challenging as regards security. Indeed, most of the suggested security measures are borrowed from the IT world and are not refit to take into account industrial systems specificity. Hence, we argue that there is still work to be done on these measures, especially Authentication and Authorization mechanisms, Segmentation and Segregation, Firewalls and IDS/IPS, in order to adapt them to industrial systems.

## References

[1] V. P. M. A. Keith Stouffer, Suzanne Lightman, , and A. Hahn, "Guide to industrial control systems (ics) security," *NIST special publication, vol. 800, no.82*, 2015.

[2] P. S. M. Pires and L. A. H. G. Oliveira, "Security aspects of scada and corporate network interconnection: An overview," in *IEEE International Conference on Dependability aof computer Systems*, 2006, pp. 127–134.

[3] "Security for industrial automation and control systems: Terminology, concepts, and models." *ISA-99 Standard 62443-1-1 (Draft2, Edit4)*, 2013.

[4] "Enterprise - control system integration part 1: Models and terminology." *ISA-dS95 Standard (Draft 14)*, 1999.

[5] "Global mag security, october 2014," *Global Security Mag*, 2014.

[6] "Security for industrial automation and control systems : Security technologies for industrial automation and control systems." *ISA TR62443-3-1 (99.03.01) (Draft1, Edit1)*, 2012.

[7] D. CSSP, "Recommended proctice: Improving industrial control systems cybersecurity with defense-in-depth strategies," *US-CERT Defense In Depth (October)*, 2009.

[8] "Classification method and key measures," *ANSSI*.

[9] "Detailed measures," *ANSSI*.

[10] M. Brändle and M. Naedele, "Security for process control systems: An overview," *IEEE Security & Privacy*, 2008. [Online]. Available: http://dx.doi.org/10.1109/MSP.2008.150

[11] T. I. JOINT TASK FORCE, "Security and privacy controls for federal information systems and organizations," *NIST Special Publication, vol. 800, no.53*, 2013.

[12] "Tofino industrial security solutions." [Online]. Available: https://www.tofinosecurity.com/why/overview

# Maliciously Manipulating a Robotic Swarm

**Ian Sargeant and Allan Tomlinson**
Information Security Group, Royal Holloway, University of London
Egham Hill, Egham. Surrey. TW20 0EX. U.K.

**Abstract**— *Most contemporary research in the field of robotic swarms assumes a benign operational environment. In our work we assume a hostile environment. We consider how swarms might be attacked and begin with a review of robotic swarm taxonomies. We then consider how a generic swarm might be attacked, and how attacks on swarms might be investigated. We conclude by presenting results of simulations of attacks that have been undertaken against swarms, based the robotic swarm taxonomies.*

**Keywords:** Robotic Swarms, Security, Attacks, Simulations

## 1. Introduction

Considerable research has been undertaken into the applications [1]–[3] and implementations of robotic swarms [4]–[8]. Much of this work assumes trust between the interacting swarm elements and trust of their operating environment. Consequently most physical implementations are realised in relatively benign conditions. Our concern, however, is with robotic swarms that might operate in hostile environments.

The aim of this paper is to demonstrate that robotic swarms can be subject to dedicated attacks, with an adversary exploiting the unique characteristics of the swarm.

To do this, we clearly need to understand the unique characteristics of robotic swarms that an adversary may exploit. We therefore review previously proposed robotic swarm taxonomies [9]–[13] to allow us to determine where the vulnerabilities in various types of swarm might lie.

We also need to understand the adversary's capabilities and the classes of vulnerabilities that an attacker may exploit. In the latter, we concentrate on vulnerabilities that are unique to swarm robotic systems. In particular, our aim is to be able to manipulate the swarm to alter it's behaviour.

In order to study how attacks may be used to manipulate the swarm, we use Netlogo simulations. This allows us to investigate, firstly if the swarm can be manipulated, and if so what resources are needed to carry out the attack.

### 1.1 Introduction to Swarm Robotics

Swarm robotic research investigates the potential uses and benefits of autonomous multi-robot systems, often taking inspiration from nature, such as swarms of bees, colonies of ants and schools of fish. The philosophy behind a robotic swarm is that a large number of relatively basic robots will work and interact with each other, and the environment [14], so as to complete a predetermined goal. It is generally agreed that a robotic swarm should have the following characteristics [14]–[18]:

- Autonomy
- Decentralised control
- Large number of members
- Collective emergent behaviour
- Local sensing
- Local communications
- Resilience to failures within the robotic swarm
- Scalable in size

In our research, the key aspects of robotic swarms are those that make them unique when compared to current information systems. In particular our interest is in the emergent behaviour of the swarm. Robotic swarms are, in theory, autonomous groups of simple swarm elements that operate remotely, with no deterministic program. Each swarm element follows a simple set of rules, interacting only locally, in order to achieve a pre-determined goal. Thus, the swarm behaviour emerges from the simple rules and local interaction. The extent to which this emergent behaviour may be disrupted is the problem that we are investigating.

At the moment, the study of swarm robotics is generally undertaken within a research environment, with proposed applications for the use of robotic swarms often being theoretical. However, the proposed numerous and diverse uses of robotic swarms, along with the the potential benefits that could be realised, drives the majority of the research. Our motivation is to understand the vulnerabilities in these systems, and how they may be mitigated, before systems are widely deployed.

There are proposed applications of robotic swarms in both civilian and military applications.

Typical civilian applications include [2], [3]:

- Maintenance tasks
- Communications provision
- Emergency response
- Use in space flights and exploration
- Use in medical procedures

Typical military applications include [19], [20]:

- Military communications
- Underwater mine clearance
- Landmine detection
- Situational Awareness

ISBN: 1-60132-433-2, CSREA Press ©

### 1.2 Introduction to the Problem

In our work, we assume that the swarm is operating in a hostile environment. And we assume that an adversary has the opportunity, and ability, to attempt to manipulate the emergent behaviour of the swarm. The goal of the adversary might be to prevent the swarm from achieving its goal, or it may be simply to slow the progress of the swarm. The latter might be an easier attack to undertake.

The overarching problem is therefore to determine whether the expected behaviour of a robotic swarm can be altered by an attacker, and if so, to what extent?

## 2. Taxonomy of Robotic Swarms

Many swarm designs have been proposed and although each may have specific vulnerabilities our goal is to identify generic vulnerabilities. Hence we need to be able to classify swarms somehow, and identify common characteristics that may be vulnerable to common types of attack.

There are various proposals defining robotic swarm taxonomies in the literature. These taxonomies are based on several factors, such as physical design considerations and collective behaviours [9]. Typical examples being the size of the swarm, communications media, coordination, control, design approach and processing abilities, and collective behaviours, such as foraging or construction [10]–[13].

Other proposals [21], [22] suggest that the attributes for consideration should be the highly dependent features of group architecture, how members deal with resource conflicts, how a collective learns and adapts to a task, how cooperation is motivated and how planning is addressed. Indeed, certain taxonomies [22] also consider the environment in which the entities are operating.

The following is an overview of the swarm taxonomies proposed to date.

### 2.1 Method Based Taxonomy

Brambilla et al. [9] suggest that swarms can be grouped according to design methods and analysis methods, as shown in figure 1.

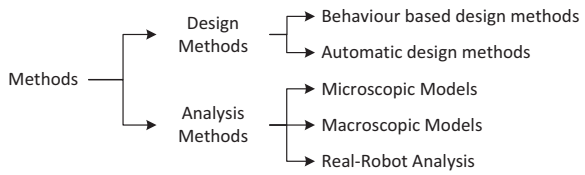



Fig. 1: Method Based Taxonomy [9]

Brambilla et al. claim that the most common design method is behaviour based, generally a bottom-up process, although it could be a top-down process, and is often based on the observations of animals. They categorise behaviour based designs into the three main categories: probabilistic finite state machine designs; virtual physics designs; and other design methods. They also suggest that automatic design methods are further classified as either evolutionary robotics or multi-robot reinforcement, and are employed with the aim of reducing the effort of developers as they do not require the intervention of the robotic swarm developer.

The authors also suggest that the analysis methods for robotic swarms can be modelled at either the individual level, the microscopic level, or the collective (macroscopic) level.

### 2.2 Collective Behaviour Based Taxonomy

In the same paper, Brambilla et al. present a second taxonomy, and categorise robotic swarms by their collective behaviours, as shown in figure 2.

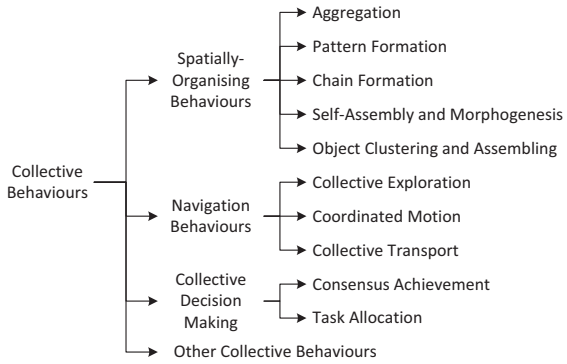



Fig. 2: Collective Behaviour Based Taxonomy [9]

The authors suggest classifying the collective behaviours by four main categories. These categories are based on how robotic swarms organise and distribute themselves; navigate and coordinate their movements; undertake decision making; and other collective behaviours that do not correspond to the above categories.

### 2.3 Swarm Behaviour Taxonomy

Cao et al. [21] also propose a taxonomy based on collective behaviour as shown in figure 3.

However, they choose different aspects to classify collective behaviour. In this taxonomy behaviour is categorised firstly by the system architecture. The remaining categories are based on: how resource conflicts are resolved; how cooperation between entities is achieved; how the swarm learns to solve problems; and how the collective entities conduct path planning from a geometric perspective.

### 2.4 Specification and Attributes Taxonomy

Another taxonomy has been proposed by Dudek et al. [12]. This classification scheme is based on a swarm's characteristics and attributes, as shown in figure 4.

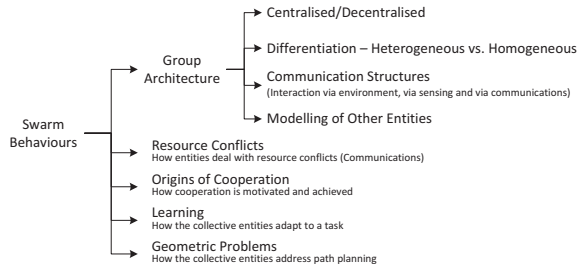



Fig. 3: Swarm Behaviour Taxonomy [21]

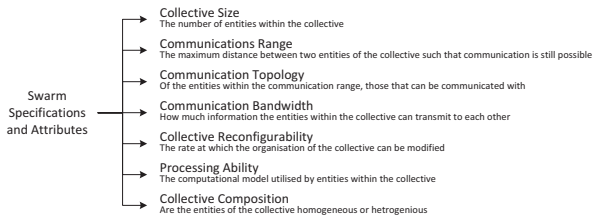



Fig. 4: Specification and Attributes Taxonomy [12]

Dudek et al. classify swarms based on the following categories. These are the size of the robotic swarm; the communications range between entities within a robotic swarm; the communication topology used; the quantity of information that can be transferred; how the swarm entities can re-configure their organisation; the processing capability that a swarm entity has; and whether the swarm's composition is homogeneous or heterogeneous.

## 3. Security Concepts

The above taxonomies give us a way to classify the unique aspects of robotic swarms where we wish to identify vulnerabilities. However, before we do so, we introduce some standard security terminology [23].

**Confidentiality:** Confidentiality is the requirement to protect information from unauthorised disclosure. Messages being transferred between elements of the swarm, might be confidential, as might data held by swarm elements.

**Integrity:** Integrity is the requirement to prevent data from being altered in an unauthorised way. This may also apply to messages between and data held by swarm elements.

**Availability:** This is the requirement for a system or data to be accessible and usable upon demand by an authorised entity. The term Denial of Service (DoS) is often used to describe a loss of availability.

**Access Control:** The requirement to allow only authorised access to a system. This generally requires identification. In a truly homogeneous swarm this may not be possible, although a group identity may be available.

## 4. The Adversary

We assume that the swarm will operate in a hostile environment. We also assume that the swarm may not always be in contact with its "owner". Thus the adversary may have access to the swarm without the "owner" being aware. These assumptions give a lot of power to the adversary but we believe they are reasonable since swarms are designed to operate autonomously and to be able to move freely.

Thus the adversary is able to freely observe and tamper with the swarm. In some cases, the adversary may be able to remove and replace elements of the swarm. Of course, any overt interference with the swarm is likely to be detected, but we assume our adversary wishes to operate covertly.

Our adversary's goal is to be able to manipulate the swarm. Ultimately the adversary would like to alter the emergent behaviour of the swarm.

## 5. Threats

Given an attacker with the capabilities described above we now consider the threats that are posed to swarm. In the following we use the term "threat" as defined by ISO/IEC 27005 [24]:

> "a potential cause of an incident, that may result in harm of systems and organisation"

This definition refers to the *potential* cause of an incident. So although many threats may exist, safeguards may be put in place to mitigate the risk of a threat being realised and exploiting vulnerabilities in the system. In order to cause harm, a threat needs to exploit a vulnerability of a asset within the system [25].

The following describes the threats to which any robotic swarm could be susceptible.

**Denial of Service:** The threat of a Denial of Service (DoS) is a threat to the availability of a system [26].

In addition to direct DoS threats to the system, a swarm is also subject to indirect threats via external environmental influences. For example, barriers placed by the adversary to impede free movement.

The DoS threat to a swarm will depend on the system design and where this design lies within the behaviour based taxonomies of Cao et al. [21] and Brambilla et al. [9]

**Masquerade:** Masquerade is the threat that an intruder or system entity illegitimately poses as, or assumes the identity of, another entity [27]. Masquerade allows unauthorised users to gain access to confidential data or greater privileges, while pretending to be legitimate user [28].

In terms of our adversary's goals, successful masquerade will allow covert operation.

**System Penetration:** The threat of system penetration is the unauthorised access to a system without posing as a legitimate entity [27]. This does not necessarily allow our adversary to operate covertly but, as with masquerade, it may enable the realisation of other threats.

**Authorisation Violation:** Authorisation violation is the use of a system by an *authorised* user for unauthorised uses, possibly abusing privileged access to resources. This can be undertaken by either an insider or external adversary [27].

In the theoretical model of a swarm, this threat may never be realised as there is no hierarchy within the swarm. However as noted by Cao et al. in their taxonomy, hierarchy may exist in practice. It is the "Group Architecture" branch of this taxonomy that may be subject to authorisation violation.

**Planting:** This is the threat that a malicious capability is planted within a system to perpetrate future attacks [27].

We are all familiar with the threats of Trojans and viruses. However, within a swarm, our adversary may be able to plant malicious *swarm entities* as described in section 4.

**Eavesdropping:** This threat is where an adversary obtains information by reading data sent between system participants, without penetrating the devices [27].

In our model, if messages are not protected, then our adversary could easily eavesdrop on the system without being detected.

**Modification of Data in Transit:** This is a threat to data in transit by either actively modifying the data or the communications process when sending data between authorised participants [27]. The threat could be to either or both user data and control information [29]. The threat is that an adversary can alter a legitimate message by deleting parts, adding extra, changing elements or reordering it [30].

Essentially this is a threat to integrity as defined in section 3 without penetrating a victim's system.

As with eavesdropping, if messages are not protected, our adversary could realise this threat without being detected.

**Misappropriation:** An attacker steals or makes unauthorised use of a service for which the service was initially unintended [30]. This is similar to authorisation violation but is not concerned with the escalation of privileges.

In our model, if this threat is realised, then the adversary could make a swarm execute a task for which it was not originally intended to undertake.

# 6. Attacks

An attack is the realisation of a threat [27]. We have considered a number of threats and how they relate to robotic swarms. Where appropriate, these threats will be realised by our adversary, and turned into attacks, to help achieve his goal of manipulating the swarm.

However, our adversary may only attack the *elements* within the swarm. The swarm, as a whole, is an abstract notion. Thus, individual swarm elements will be attacked, in order to manipulate behaviour of the swarm.

In other work [31] we proposed that there are four different types of attack that can be used to manipulate a swarm regardless of the swarm's specific implementation. These are summarised below.

**Manipulate an element's goal:** If attribute $a_0$ of a swarm element $s_x$ defines it's goal, then manipulating $a_0$ (e.g. switch from "search" to "defend") can ultimately manipulate the swarm.

**Manipulate an element's behaviour:** A swarm element will have a number attributes, (e.g. separation, speed) If an adversary is aware of the effect that these attributes have on the behaviour of $s_x$, then he can, directly or indirectly, manipulate the attributes and ultimately manipulate the swarm.

**Manipulate the environment:** An adversary may alter the environment in order to manipulate the swarm.

**Manipulate communications via the environment:** The adversary may be able to modify the environment in order to inhibit communications or, in the case of stigmergy, to manipulate data in transit.

The above may be considered as generic attacks on a swarm. Since these are attacks on swarm elements there are numerous attack vectors that can be used. The specific attack vector used will depend on the technology used to implement the swarm, and to some extent, where the swarm lies in the Group Architecture class of Cao et al.

In the following, we list some of the more specific attack vectors that can be used against swarms. All of these attack vectors may be uses to implement one or more of the generic attack types described above.

- Replay: A replay attack is an attack in which a past message is played back to the same recipient [32]. In some cases it may be possible to modify parts of the message before replaying. Messages between swarm elements or from the environment may be replayed.
- Physical Tampering: We assume that an adversary may capture a swarm element. Having done so, the adversary my be able to physically attack the device to extract or modify data. This is different to traditional computing,

where physical protection of assets can be achieved by restricting physical access to devices.

- Software: Software or firmware attacks are concerned with modifying code, and exploiting vulnerabilities [33]. Software attacks on a swarm can be realised in several ways, such as via physical tampering, planting, or from within the supply chain.
- Communications: Attacks may be made on communications between swarm elements and between elements and the environment. Eavesdropping and modification of data are obvious attacks that can be made here. This becomes especially pertinent if the adversary is able to masquerade as a genuine swarm element, perhaps after capture and re-introduction to the swarm. Alternatively, the adversary could carry out these attacks from a static position in the area that the swarm is operating. An attacker may also block communications (DoS).
- Reconnaissance: In order to prepare for an attack, an adversary may undertake reconnaissance of a swarm in order to gain intelligence. This can be done by either traffic analysis or visual observation of the swarm.

# 7. Simulating an Attack

In order to test our hypothesis that we could manipulate a swarm by tampering with a few swarm elements we carried out some simulations using Netlogo 5.0.2.

## 7.1 Cooperative Navigation

We chose to simulate the "Cooperative Navigation" swarm proposed by Ducatelle et al. in 2013 [34]. The algorithm provided in this work was sufficient for us to design our simulation. The model was first simulated in a benign environment to ensure that our results were comparable.

The original research presented various options for cooperative navigation. This included a *searcher* attempting to find a *target* using information gained from other robots to aid in navigation.

From the *Method Based* taxonomy of Brambilla et al. this implementation would be considered as a *finite state machine* within the *Behaviour Based* design methods category. Many swarm implementations fall into this category.

In the *Collective Behaviour* based taxonomy of Brambilla et al. this would be classed as *Collective Exploration*, within *Navigational Behaviours*

In the *Group Architecture* class proposed by Cao et al. this activity would be classed as a *Communications Structure*.

Cao et al. may also consider this as a *Geometric Problem*, since the swarm is assisting in path planning. However, the collective entities are only passing on information to assist in path planning and not actually planning the path themselves.

In this swarm, the searcher and all entities contain stored information relating to *distance to the target* and the *information's age*. The target provides a count that is used to determine the age of the information received by each swarm element regarding it's distance to the target.

After a set time period, the target updates its count. Any entities from the swarm that are within the communication range of the target update their knowledge of their own distance to the target and the targets current count (for the information age). This data is then distributed through the swarm by sending messages to swarm members that are in range of each other. The receiving entity modifies its distance to the target if the information received is fresher than its stored information. The swarm does not act on this data, other than to forward the information on.

The searcher is also able to receive the distance to target and information age. If the data received by the searcher is "better" than its stored information then the searcher will update its data and the travel towards the position of the swarm entity that provided the information. By "better" we mean that either the information is fresher or if the information is of the same age but the distance to the target is shorter than the searcher's own data. The process continues until the searcher reaches the target.

If a searcher does not receive any information, or the swarm entities do not have any fresher information, then the searcher has two options. It can either stop and wait at its location for fresher information, or it can then travel in a random direction for a random distance, or until it receives fresher information.

## 7.2 Attacking Cooperative Navigation

Our initial attacks were Denial of Service. This was undertaken by the attacker manipulating the communications provided via the environment by means of jamming the signals. The jamming attacks are limited in both the frequency of the jamming event and the size of the area affected by the jamming event. The area affected was simulated to be the same size as the area that an individual robotic swarm entity could communicate within. Therefore communications were still possible between swarm entities that outside the jamming area.

Further experiments were then carried out where malicious entities manipulated the information being communicated between robotic swarm entities. This included manipulating distance to target and freshness of data values, in order to influence the robotic swarm entities.

## 7.3 Simulation Results

The effects of the DoS attacks can be seen in the figures 5 and 6 below. In these attacks, ten malicious entities are randomly positioned within the operating environment. The malicious entities, once dispersed, are static, have a 1% chance of undertaking a jamming event, and will jam for 5 time units.

Figure 5 shows how the effectiveness of an attack by various numbers of malicious entities, is reduced as the number

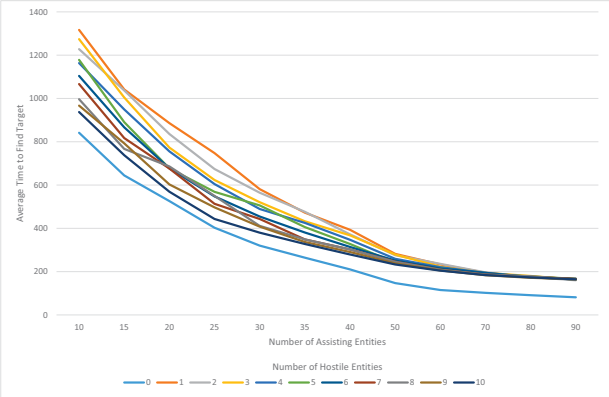



Fig. 5: DoS on a Robotic Swarm

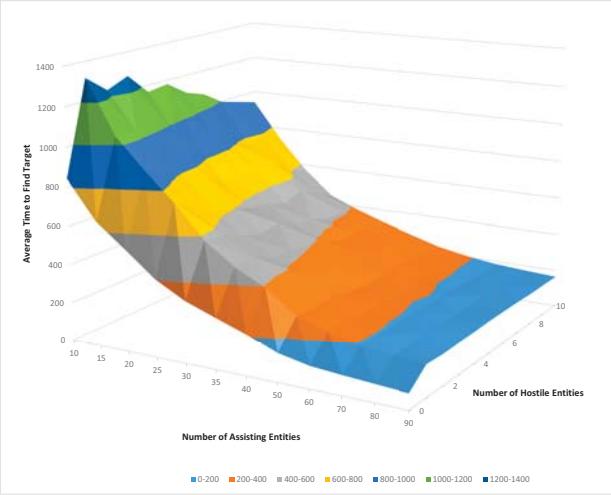



Fig. 6: Combined View of DoS on a Robotic Swarm

of genuine swarm entities increases. An attacker could use this information to optimise the number of malicious entities deployed for greatest effect.

Figure 6 demonstrates that this type of attack used can have unusual results. We designed the malicious entities to operate covertly, acting as part of the robotic swarm when not engaged in the DoS attack. Thus, when not attacking, the malicious entities actually assist the searcher in achieving its goal. Of course the adversary could counter this by increasing the attack period by increasing the likelihood of jamming events, or by operating a little less covertly. In the latter case, the malicious entities would monitor the swarm, and choose when to attack, but would not act as part of the robotic swarm.

### 7.4 Discussion

From our initial results we can see firstly, that we are able to manipulate the effectiveness of a swarm by introducing a small number of rogue elements. The simulation gives us a tool to study the effectiveness of an attack and investigate how to optimise the number of rogue elements.

Secondly, the simulation uncovers unexpected outcomes. Although, with hindsight, some outcomes are understandable, the non-deterministic nature of the swarm makes simulation a useful tool for both developer and adversary.

From this we can see that although the attack vectors are not new, their unique characteristics make swarms react in subtly different ways when compared with the same attack on traditional information systems.

The swarm chosen to simulate fits into several categories in the taxonomy. In theory we should now be able to choose swarms from similar categories and expect similar results. Alternatively we could investigate different classes of design.

We are currently running DoS simulations that utilise mobile jammers, in order to complete legacy style attacks. We are also running attacks based on the modification of data in transit. These attacks are aimed at tampering with data communications to manipulate the received distance to target and the information age. In other words, we are tampering with the attributes stored by individual swarm elements in order to manipulate the swarm.

## 8. Conclusion

In conclusion, this paper provides the researcher with an overview of various robotic swarm taxonomies and a understanding of various types of threats and attack methodologies. This is the basis for investigating vulnerabilities in different classes of robotic swarm systems.

From our initial experiments, we have demonstrated that robotic swarms can not only be attacked by traditional methods, but also that the attacks can utilise the unique characteristics of robotic swarm.

Our work is continuing to investigate the effectiveness of attacks and how the swarm may be able to defend itself against such attacks.

## References

[1] P. Scerri, D. Pynadath, L. Johnson, P. Rosenbloom, M. Si, N. Schurr, and M. Tambe, "A prototype infrastructure for distributed robot-agent-person teams," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '03. New York, NY, USA: ACM, 2003, pp. 433–440. [Online]. Available: http://www.cs.cmu.edu/~pscerri/papers/rap-aamas03.pdf

[2] S. Subchan, B. White, A. Tsourdos, M. Shanmugavel, and R. Zbikowski, "Pythagorean Hodograph (PH) Path Planning for Tracking Airborne Contaminant using Sensor Swarm," in *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE*, 2008, pp. 501–506.

[3] H. Woern, M. Szymanski, and J. Seyfried, "The I-SWARM project," in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, Sept 2006, pp. 492–496.

[4] A. Martinoli, K. Easton, and W. Agassounon, "Modeling swarm robotic systems: A case study in collaborative distributed manipulation," *Int. Journal of Robotics Research*, vol. 23, pp. 415–436, 2004. [Online]. Available: http://www.kjerstin.com/pubs/04_AMKEWA_IJRR.pdf

[5] J. Fredslund and M. J. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 837–846, Oct 2002. [Online]. Available: http://robotics.usc.edu/publications/downloads/pub/47

[6] T. Balch and R. C. Arkin, "Behavior-based formation control for multi-robot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 926–939, 1997. [Online]. Available: http://www.cc.gatech.edu/ai/robot-lab/online-publications/formjour.pdf

[7] C. M. Cianci, X. Raemy, J. Pugh, and A. Martinoli, *Swarm Robotics: Second International Workshop, SAB 2006, Rome, Italy, September 30-October 1, 2006, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. Communication in a Swarm of Miniature Robots: The e-Puck as an Educational Tool for Swarm Robotics, pp. 103–115. [Online]. Available: http://infoscience.epfl.ch/record/100015/files/44330103.pdf

[8] B. P. Gerkey and M. J. Mataric, "Sold!: auction methods for multirobot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, Oct 2002. [Online]. Available: http://cres.usc.edu/pubdb_html/files_upload/10.pdf

[9] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11721-012-0075-2

[10] G. Dudek, M. R. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996.

[11] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11721-012-0075-2

[12] G. Dudek, M. Jenkin, and E. Milios, "A taxonomy of multirobot systems," *Robot Teams: From Diversity to Polymorphism*, pp. 3 – 22, 2002.

[13] G. Dudek, M. R. M. Jenkin, and D. Wilkes, "A taxonomy for swarm robots," in *Proceeding on the IEEE/RSJ International Conference on Intelligent Robotic Systems*. IEEE, 1993, pp. 441–447.

[14] E. Şahin, S. Girgin, L. Bayindir, and A. E. Turgut, "Swarm robotics," in *Swarm Intelligence: Introduction and Applications*, ser. Natural Computing Series, C. Blum and D. Merkle, Eds. Springer, 2008, pp. 87–100.

[15] A. F. T. Winfield and J. Nembrini, "Safety in Numbers: Fault Tolerance in Robot Swarms," *International Journal on Modelling Identification and Control*, vol. 1, no. 1, pp. 30–37, 2006.

[16] A. Yamashita, T. Arai, J. Ota, and H. Asama, "Motion planning of multiple mobile robots for cooperative manipulation and transportation," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 2, pp. 223–237, 2003.

[17] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 2, pp. 220–240, 1998.

[18] M. Gerla, Y. Yi, K. Xu, and X. Hong, "Team communications among airborne swarms," in *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, vol. 3, March 2003, pp. 1303 – 1312.

[19] E. Chapman and F. Sahin, "Application of swarm intelligence to the mine detection problem," in *SMC (6)*, 2004, pp. 5429–5434.

[20] Y. C. Tan and B. Bishop, "Evaluation of robot swarm control methods for underwater mine countermeasures," in *System Theory, 2004. Proceedings of the Thirty-Sixth Southeastern Symposium on*, 2004, pp. 294–298.

[21] Y. Cao, A. Fukunaga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, no. 1, pp. 7–27, 1997. [Online]. Available: http://dx.doi.org/10.1023/A%3A1008855018923

[22] R. C. Arkin, T. Balch, and E. Nitz, "Communication of behavorial state in multi-agent retrieval tasks," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 588–594.

[23] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, ser. Discrete Mathematics and its Applications. Boca Raton, Fl: CRC Press, 1997, vol. 6. [Online]. Available: http://www.cacr.math.uwaterloo.ca/hac/

[24] *BS ISO/IEC 27005:2011 – Information technology – Security techniques – Information security risk management*, British Standards Std.

[25] *BS ISO/IEC 13335-1 Information Technology - Security Techniques - Management of Information and Communications Technology Security*, British Standards Std.

[26] M. Whitman and H. Mattord, *Principles of information security*. Cengage Learning, 2011.

[27] W. Ford and M. S. Baum, *Secure electronic commerce: building the infrastructure for digital signatures and encryption*. Prentice Hall PTR, 2000.

[28] W. Naik Bhukya, G. Suresh Kumar, and A. Negi, "A study of effectiveness in masquerade detection," in *TENCON 2006. 2006 IEEE Region 10 Conference*, Nov 2006, pp. 1–4.

[29] S. Kelly and T. C. Clancy, "Control and provisioning of wireless access points (CAPWAP) threat analysis for IEEE 802.11 deployments," 2009, Network Working Group, Internet Engineering Task Force.

[30] K. Scarfone, D. Dicoi, M. Sexton, and C. Tibbs, "Guide to securing legacy ieee 802.11 wireless networks," *NIST Special Publication*, vol. 800, p. 48, 2008.

[31] I. Sargeant and A. Tomlinson, "Review Of Potential Attacks On Robotic Swarms," in *IntelliSys 2016*. IEEE, 2016.

[32] L. Gong, "Variations on the themes of message freshness and replay-or the difficulty in devising formal methods to analyze cryptographic protocols," in *Computer Security Foundations Workshop VI, 1993. Proceedings*, Jun 1993, pp. 131–136.

[33] T. Roosta, S. Shieh, and S. Sastry, "taxonomy of security attacks in sensor networks and countermeasures," in *In The First IEEE International Conference on System Integration and Reliability Improvements. Hanoi*, 2006, pp. 13–15.

[34] F. Ducatelle, G. A. Caro, A. Förster, M. Bonani, M. Dorigo, S. Magnenat, F. Mondada, R. O'Grady, C. Pinciroli, P. Rétornaz, V. Trianni, and L. M. Gambardella, "Cooperative navigation in robotic swarms," *Swarm Intelligence*, vol. 8, no. 1, pp. 1–33, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11721-013-0089-4

# Reducing False Alarms Caused by Wind Effect in Automatic Security Perimeter System

**Hiesik Kim**[1], **Odgerel Ayurzana**[2]
[1]Department of Electrical and Computer Engineer, University of Seoul, Seoul, Korea
[2]Department of Electronics, Mongolian University of Science and Technology, Ulaanbaatar, Mongolia

**Abstract -** *False alarms due to wind effect on the perimeter security system need to be reduced. Sensor cables are fastened to existing security fences. Main problem of the system is a false alarms generated by the strong wind. Input signals receiving from sensor cables are processed and analyzed by using the DSP (TMS320F2812) microcontroller. The system collects signal amplitude, duration time and frequency spectrum to distinguish the wind effect from intrusion signal. The result gives the real or false alarm. Frequency analysis was done for each duration of N=128 samples of input signal was using special algorithm of the fast Fourier transform through DSP microcontroller. The security system has been tested at border lines of Mongolia. A detection rate after apply the developed algorithm for reducing false alarms was improved up to 94%~95% of enough accuracy for real field application.*

**Keywords:** Triboelectric, Electrostatic Charge, Security Fence, Automatic Detection, Intrusion Detection, False Alarm

## 1 Introduction

The automatic electrical security systems with intrusion detection sensors become more critical problem for any security organization and countries. Many strategically important factories and facilities have been built recently by using rapid industrial it technology development. Security systems were developed with the advanced technologies. They are in high demand to ensure reliable security of industry buildings and special governmental institutions.

The physical security system has been developed from the first generation to the third generation by using the latest modern technologies. The first generation was dependent on the barbed wire fence and guards' eyes inspection. The second generation fence was based on electric fences and CCTV (Closed-circuit Television) cameras. The third generation includes a surveillance system based on GPS (Global Positioning System), cameras and wireless communication system, an alarm system with various sensor fences, an access control system from outside gates to the inside of the buildings. The systems are integrated into a central security control server. The purpose of the physical protection system is to detect an unauthorized intruder whereas the perimeter intrusion detection sensor fence detects the intruder right away by a surveillance monitoring system which covers before and after the intrusion. Also at the same time it prevents the intruders by a sensor at the earliest time. There are several kinds of fences in the second generation such as a taut-wire, a vibration, an electromagnetic induction sensor and a fiber optic cable and a micro wave, and an electric charge sensor in the third generation [1, 2]. The minimal charge is multiplied by semiconductor to be detected. It has the unique characteristics as charge amplifier in a circuit containing a tunnel diode biased for charge detection [3]. In paper [4], there is a polyelectric polymer sensor with an integrated charge amplifier.

The fence security systems with the fiber optic cable can define certain location of an intrusion. But the system cost and maintenance expense are higher than other similar systems [5, 6].

The first prototype device design of the security system based on the triboelectric effect was developed and investigated in the study [7]. The electric charge sensor fence was developed as the prospected system in this field. It is the first passive sensor in the world. It uses the special algorithm of electric charge displacement that is based on friction electromotive force change of the passive sensor cable. The electric charge technique is based on the triboelectric effect. If we rub two different types of materials with each other, the electrostatic charges are generated that are called the triboelectric effect. The physical protection security system can be realized by triboelectric effect. A simple telecommunication cable is used in the protection systems instead of the sensor transducer that is fastened to the various types of perimeter fences. Electrostatic charges are generated, when external force is applied to the sensor cable.

The ASM (Analog Sensing Module) [8, 9] detects the generated minimal charges and rings an alarm. The ASM doesn't generate any electric signals and electromagnetic fields. It only reacts against the change on electrostatic charge in specific range and is not affected by any external factors. Therefore there are no false alarms and perfect probability detection. Also the security detector does not generate the false alarm signals on and after exposure to the outdoor environment factors including humidity, rain, wind, fog, dust etc. the sensor cable reacts against the forced impact weighing 8-20 kg on the fence. It does not detect little forces including

small animals hit the fence. The system sensitivity can be regulated by switches. Electric charge sensor fences are not harmful to the human body. These types of security systems have much more advantages than any other systems. It has simple installation and easy maintenance and is (highly affordable) more economical than any other systems.

# 2 System design and its solution

## 2.1 System structure

Electrostatic charges are generated between the cable conductors and isolator when external force and impact are created by the intruders. Simple shielded 15 pair of telecommunication cable is used instead of the sensor transducer for sensing the external force and impact. Sensor cable is not connected to the power source. The lengths of sensor cable are limited up to 1000m. Fig. 1 shows the main operation diagram of the system.

Coaxial cable is used for transferring generated minimal electrostatic charge between the sensor cable and control device. Because of the very high input impedance of the charge amplifier, the sensor must be connected to the amplifier input with low-noise cable [9]. This cable is specially treated to minimize triboelectric noise which is generated within the cable due to physical movement of the cable. The coaxial cable is necessary to affect an electrostatic shield around the high impedance input lead, precluding extraneous noise pickup.

A control device is connected to the data center through the TCP/IP protocol. The control device includes the TCP/IP module. The data center registers fence alarms and control device's conditions when some problems are occurred on the security fence. For example, there are generated alarms when the sensor or coaxial cable is cut or short.

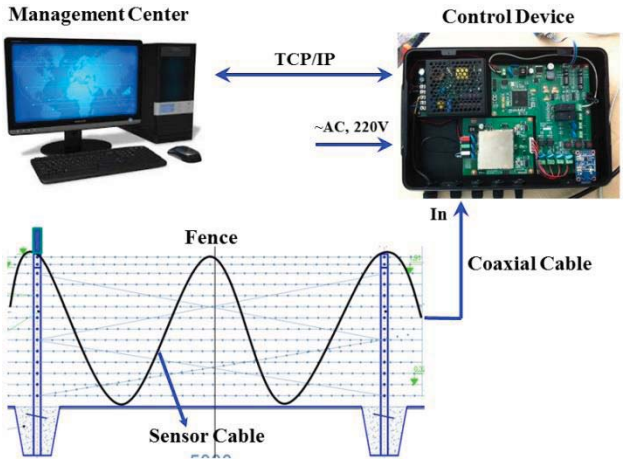



Fig. 1. Main operation diagram of the system

Installation configuration of the sensor cable is dependent on the fence type, structure, and size, height, and installation weather conditions. System sensitivities can be adjusted by hardware method in the control device. Also sensitivities can be adjusted by software in the monitoring application program at the data center. Sensitivity is adjusted by less when sensor cable is fastened on flexible and moving fences. Conversely, sensitivity is adjusted by high when cable is fastened on rigid and less moving fences. The control device controls up to two security zones. Each zone are covered with 500m sensor cable.

## 2.2 Operation principle of a control device

The control device contains two main parts named as ASM (Analog Sensing Module) and SCM (Sensitivity Control Module).

The ASM [7, 8, 9] detects the electrostatic charges on the passive sensor cable by intruder and raises the alarm. The ASM contains a charge sensitive device, voltage amplifier, signal shaping, filtering, and comparator. The charge sensitive device consists of the charge preamplifier and filters.

The SCM is the digital part of the control device that processes and analyses analog signals from a ASM using by TMS320F2812 32 bit DSP (Digital Signal Processing) microcontroller. The input signal of the sensor cable is fluctuated due to outside environment effects (strong wind, storm) then the SCM analyses all conditions and adjusts sensitivity automatically. For instance SCM reduces sensitivity during strong wind and increases sensitivity during low wind.

Also SCM analyses the input signal amplitude, duration time and frequency. These parameters are changed due to the wind effect. After analyzing input signal, system decides which alarms are real or false. Each N=128 samples of input signal were analyzed special algorithm of the discrete Fourier transform by using equation (2.1).

$$X(k)=\sum_{n=0}^{N-1} x(n)*e^{\frac{-j2\pi k}{N}n};0\le k\le N-1 \qquad (2.1)$$

x(n): Input signal
N=Count of sample (128)

The 1Hz±0.5Hz frequency is generated in the control device when someone forces by 8~20 kg impact on the fence. This is an invasion frequency. The band bass filter is designed in the ASM of control device [8, 9]. Other frequencies are generated due to strong wind effects.

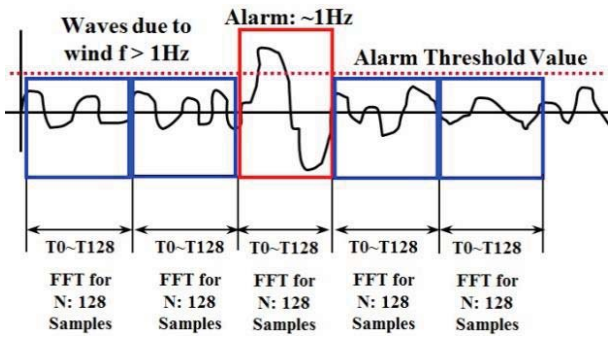



Fig. 2. Condition of fast Fourier transform in input signal

Fig. 2 shows alarm condition of the system. Input analog signal of the sensor cable is sampled by 10ms time step in each N=128 samples using by ADC of the control device. Frequencies are defined every T=1ms*128=1.28s with the fast Fourier transform.

Real wind configuration is shown in Fig. 3. False alarms are generated when signal amplitude of strong wind exceeds a reference value.

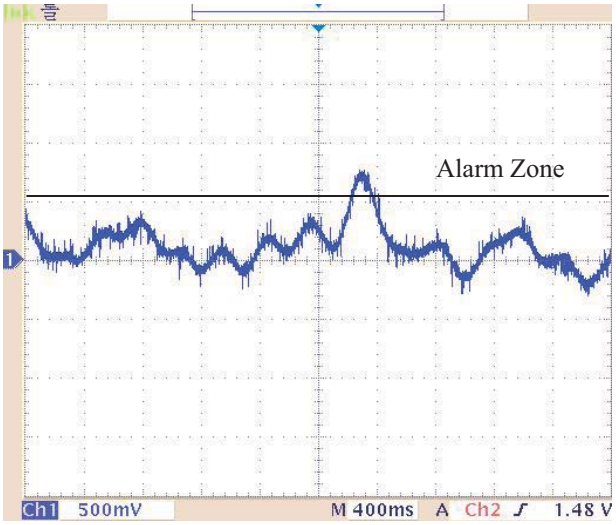



Fig. 3. Noise Signal Generated by Wind Effect on the Electrical Fence

Fig. 4 shows algorithm for reducing false alarms. This algorithm reduces false alarms due to strong wind effect. ADC of control device processes N=128 sample data by 10ms steps. An algorithm reviews and analyses frequency value using FFT, maximum and minimum value of amplitudes.

- ✓ Vmax: Maximum value of amplitude in each duration time of 1.28ms
- ✓ Vmin: Minimum value of amplitude in each duration time of 1.28ms
- ✓ Fmax: Maximum frequency in each duration time of 1.28ms by FFT
- ✓ Vpw: Value of plus wind level that is regulated by DIP switch on control device
- ✓ Vmw: Value of minus wind level that is regulated by DIP switch on control device
- ✓ Vpth: Plus threshold value
- ✓ Vmth: Minus threshold value

Then all of these values are compared to the reference values. After that system decides which alarms are real or false.

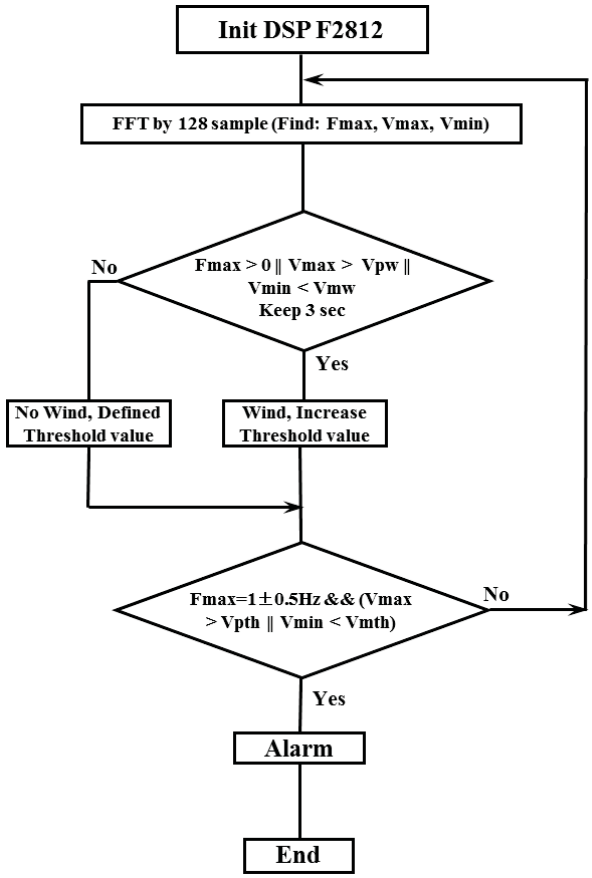



Fig. 4. Algorithm of the DSP Program for Reducing False Alarms

A detection rate was 85~88% before implementing algorithm for reducing false alarms. Detection rate of the system is improved up to 94~95% after implementing that algorithm. That means system does not alarm during strong wind. All data of the wind effect is saved to data center. We have to improve this algorithm using by saved data.

# 3 Experiment and results

## 3.1 Experiment area and conditions

The perimeter security system has been tested at the border of Mongolia (2014.11.09 ~ to present). The area has special weather conditions. In the north of Mongolia it is very cold and snowstormy during winter seasons. Also there is strong wind during spring and fall seasons.

Fig. 5 shows installation of sensor cables in the selected special zones of experiment area. Sensor cable is fastened to the border fence by sine configuration. Sensor cable configuration is depended on fence types.

Fig. 5. Installation of Sensor Cable in the Testbed Experiment in Real Field

Two separated zones for testing security system were set up on the barbed fence. This type of fence is flexible and easy moving that is very sensitive in wind effect.

Experiment has been done two-three times in a day. Operators have pull and push barbed fence then alarms are generated on the data center. Monitoring application program registers all generated alarms and system conditions.

Table 1 shows some registered alarms in the data center from 2014.11.10 to 2015.04.25. As shown in experiment results, there are no false alarms when there are windless days. Alarms are generated when a dog and a cow touches the fence. Also some alarms are generated when bevy of pies and crows seat and fly on the barbed wire of the fence. This type of alarms are normal operation of system. But some false alarms are generated during snowstorm in 2015.01.05,06.

Table. 1. Registered alarms

| Alarm time | Alarm zone | Alarm reason | Operator |
|---|---|---|---|
| 11/15/2014 12:3 | 1 | Test | P.Davaadorj |
| 11/28/2014 12:3 | 2 | Test | S.Chuluun |
| 12/16/2014 8:29 | 2 | A dog touch | G.Olzii |
| 12/27/2014 9:03 | 1 | Test | P.Davaadorj |
| 1/5/2015 2:06 | 2 | Snowstorm | B.Shinebayar |
| 1/5/2015 12:19 | 1 | Snowstorm | B.Shinebayar |
| 1/6/2015 14:12 | 2 | Snowstorm | G.Olzii |
| 1/8/2015 11:58 | 1 | A cow touch | B.Shinebayar |
| 1/13/2015 14:05 | 1 | A bevy of pies | E. Erdene |
| 1/16/2015 11:29 | 2 | A bevy of crows | S.Chuluun |
| 1/18/2015 15:13 | 1 | A cow touch | B.Shinebayar |
| 2/8/2015 11:58 | 2 | Test | B.Shinebayar |
| 2/23/2015 14:05 | 1 | A dog touch | G.Olzii |
| 3/11/2015 11:29 | 2 | Test | P.Davaadorj |
| 3/28/2015 15:13 | 1 | A cow touch | B.Shinebayar |
| 4/17/2015 13:35 | 1 | Test | S.Chuluun |
| 4/25/2015 14:05 | 2 | Test | G.Olzii |

## 3.2 Monitoring application program

Fig. 6 shows a screenshot of the monitoring application program.

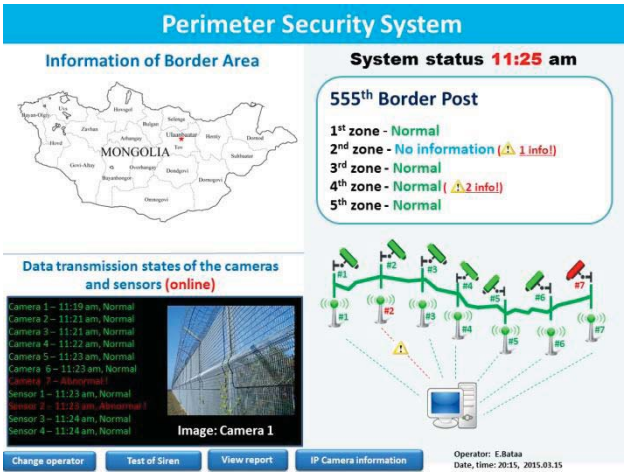



Fig. 6. Screenshot of monitoring program

All log data is stored to the data center. Monitoring program receives all zones information by real time from control device and displyas these states. For example there are intruders' alarms, sensor and coaxial cable is cut or short, control device's cover open. A report can be printed and viewed with many options.

# 4 Conclusions

The perimeter security system was developed and tested in real field experiment under the severe weather conditions of Mongolia. Main problem of the perimeter security system was false alarms. False alarms are generated due to strong wind effect. In order to reduce false alarms, special algorithm was researched and implemented in the system. As shown in experiment results, alarms are generated when a dog get in under barbed fence and a cow touches the fence. Also some alarms are generated when bevy of pies and crows seat and fly on the barbed wire of the fence. These types of alarms are normal operation. But some false alarms are generated when snow is stormed continuously.

The detection rate of the invasion is improved up to 95% as seen from experimental data of the last 5 months. To increase detection rate, algorithm for reducing false alarm has to be improved. Also to improve detection quality, IP camera has to be introduced to the system in the further work.

The sensor cable is frozen when outside temperature is decreased to -30C. So fence sensitivity is decreased. In this case we have to increase sensitivity by 1-2 steps in winter season.

# 5 Acknowledgment

This research was financially supported by the Project No. 201304221024 "Security Fence Detection Accuracy of Cable

Friction Electricity Sensor System" of the University of Seoul and by the project No. 201501262003 "Sensitive and Errorless Electrical Security Fence by Using Minimal Friction Electricity in Coaxial Cable" of the National Research Foundation of Korea and by the Project No. COR_01/2015 “High Sensitive and Errorless Security Fence System” of the Mongolian Foundation for Science and Technology.

## 6 References

[1] NISE East Electronic Security Systems Engineering Division "Perimeter Security Sensor Technology Handbook" North Char Leston, South Carolina, 1997.

[2] Garcia, L. Mary “The Design and Evaluation of Physical Protection Systems” Woburn, MA: Butterworth-Heinemann, 2001

[3] Robert J. Locker, "Minimum Charge Detection using Selected Tunnel Diodes and Charge Multiplying Semiconductors" IEEE Transactions on Nuclear Science, pp. 382-388, February 1966.

[4] D. Setiadi, H. Weller, T. D. Binnie "A Polyelectric Polymer infrared sensor array with charge amplifier readout" Elsevier Science Sensors and Actuators, pp. 145-151, February 1999.

[5] ZINUS Co., Ltd "Fiber Optic Mesh Security Fence" September 2006.

[6] Fiber SenSys Co., Ltd "Fiber Optic Intrusion Detection System" 2006.

[7] Odgerel, Ayuzana, Lee Young Chol “Security System Design based on the Triboelectric Effect” The 7th Conference on National Defense Technology, Korea University, Seoul, Korea, July 07~08, 2011

[8] Hiesik Kim, Seok Jin Yun, Odgerel Ayurzana “Implementation of Intrusion Monitoring System to Operate Optimim by using Electronic Security Fence of Friction Electricity Sensor” ICS-2015, International Conference, Gannin City, South Korea, Apr 23~24, 2015

[9] Odgerel Ayurzana, Hiesik Kim “Minimal Electric Charge Detection Device for Fence Security Systems”, MUSTAK-2015, Ulaanbaatar, Mongolia, Aug 20~21, 2015

ISBN: 1-60132-433-2, CSREA Press ©

# SESSION

# SOFTWARE TOOLS AND SYSTEMS + NOVEL APPLICATIONS

## Chair(s)

**TBA**

ISBN: 1-60132-433-2, CSREA Press ©

ISBN: 1-60132-433-2, CSREA Press ©

# Design Error Injection, Simulation, and Diagnosis Using FPGA-Based Boards

Mohammad Mahmoud, Hesham M. Elmaghraby, and Hussain Al-Asaad
Department of Electrical and Computer Engineering
University of California—Davis

*Abstract* — this paper presents a VHDL-based CAD tool that integrates design error injection, simulation, and diagnosis for logic circuits. The tool uses an FPGA-based board to inject error models in the design and compute the error free and erroneous signatures of internal lines. The signatures are later used for detection and diagnosis of errors within the design. Several experiments were conducted to demonstrate the capabilities of the tool. The obtained results demonstrate that the tool was able to detect then locate the source faulty node(s) within the design.

*Keywords— Design error simulation, design error diagnosis, automatic test equipment, FPGA-based boards.*

## I. Introduction

Fault simulation consists of simulating a circuit's behavior in the presence of faults [1]. By comparing the response of the circuit to that of the fault-free response using the same test set, we can detect the possible existence of faults. Fault simulation has many applications, such as test set evaluation, fault-oriented test generation, fault dictionaries construction, and analysis of circuit operation in the presence of faults. Recently, simulation-based design verification have received much attention [2][3]. An important part of the design verification process is design error simulation. Developing efficient design error simulators is of great interest since the existence of such simulators may benefit many areas such as test generation for design errors and design error diagnosis and correction [4][5]. To speed up the design error simulation, we can do the simulation using FPGA-based boards that are becoming readily available in these days. This form of simulation using hardware is often called emulation.

In this paper, we develop a CAD tool that can perform an efficient error/fault simulation using an FPGA-based board. The tool will generate the model for the design to be simulated in the presence of faults. It will then map it to the FPGA-based board where the actual test generation and response analysis is performed. The signatures of internal nodes in the design will then be extracted back from the FPGA to the tool to be used later for design error diagnosis.

The rest of this paper is organized as follows. Section II discusses the design errors and logical faults models that can be handled by the tool. Section III discusses the details of the tool. Section IV discusses the experimental results on some circuits. Finally, we draw some conclusions and suggest directions for further research in Section V.

## II. The Fault Model

Many types of faults and design errors have been classified in the literature [1][2][3][6]. These error types are not necessarily complete, but they are believed to be common in the lifetime of a digital system.

According to experiments reported in [7], the most frequent error made in manual design is gate substitution error (GSE), accounting for around 67% of all errors. Gate substitution refers to mistakenly replacing a gate G with another gate G' that has the same number of inputs as G. We represent this error by G/G'. For gates with multiple inputs, a multiple-input GSE (MIGSE) can have one of six possible forms: G/AND, G/NAND, G/OR, G/NOR, G/XOR, and G/XNOR. Each multiple-input gate can have five MIGSEs. For example, all MIGSEs can occur on an AND gate except G/AND which is not considered an error. For gates with a single input, i.e., buffers and inverters, a single-input GSE (SIGSE) can have one of two possible forms: G/NOT and G/BUF. Each single-input gate can have only one SIGSE. It has been suggested that most GSEs can be detected by a complete test set for SSL faults [2].

In this paper, we mainly consider MIGSE errors in the design although our method can be applied to any other fault/error models.

## III. Details of the Tool

Our tool uses the VHDL language to model the design and the associated hardware that need to be mapped to the FPGA board. The test set used in our tool is the exhaustive test set generated in hardware via a counter and applied to the circuit under test (CUT).

Our tool is based on getting the signature of each node in the circuit when applying exhaustive test pattern generation to the CUT. After injecting the modeled fault, we receive the response from the CUT and display the

ISBN: 1-60132-433-2, CSREA Press ©

signatures and generate all necessary signals for the CUT. In fact, our software provides the signatures of all nodes simultaneously in one measurement. We are in fact using signature analysis which is a technique pioneered by Hewlett-Packard Ltd., that detects errors in data streams caused by hardware faults in digital systems. Signature analysis uses a unique data compression technique which reduces long data streams into digital hexadecimal signatures.

After getting the signatures, we can compare the CUT response with that of the golden circuit to locate the source of the faulty node.

Our first experiment targeted the simple combinational logic circuit c17 (Shown in Fig. 1) that is supplied by the ISCAS-85 benchmark suite. We built our error simulator for c17 using Mentor Graphics & Quartus II tools. To inject the MIGSE error model in the design and getting the signature of each node of the circuit, we replace each NAND gate in c17 with other 5 other gate types.

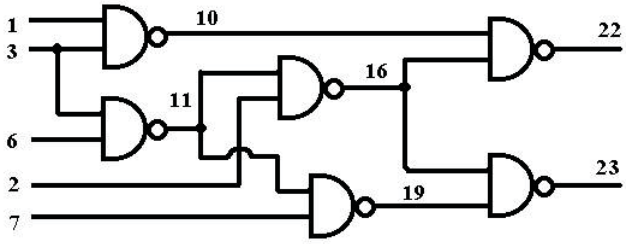



Fig. 1. The c17 circuit.

We developed a library package of MIGSE models which contains 6 types of gates (G/AND, G/NAND, G/OR, G/NOR, G/XOR, and G/XNOR) to be injected to the CUT. Depending on the selection of the control gate at a time, the fault model of the gate will be injected as shown in Fig. 2.

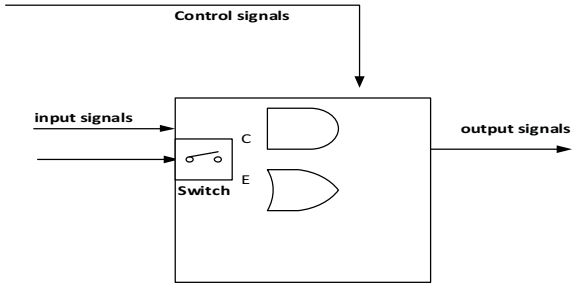



Fig. 2. Synthesizable fault model for MIGSE.

The first phase of the tool is the signature generation stage (shown in Fig. 3). In our work, we used exhaustive test pattern generation for the CUT .We used counter for flipping the gates and a decoder for selecting the control through router which assigns the selected control gate at any given time. There is flag generation assigned for starting and ending the whole process of flipping gates. After the mapping of the design to the FPGA, each gate will be flipped 5 times. The output signatures of the CUT (192 bits) will be stored in the register. The 192 bits are composed of the signatures of fault free circuit and the signatures in the presence of 30 MIGSEs on the 6 NAND gates (5 flips on each gate). The internal lines monitored are 6 and hence we have (1+30)*6 = 192 bits.

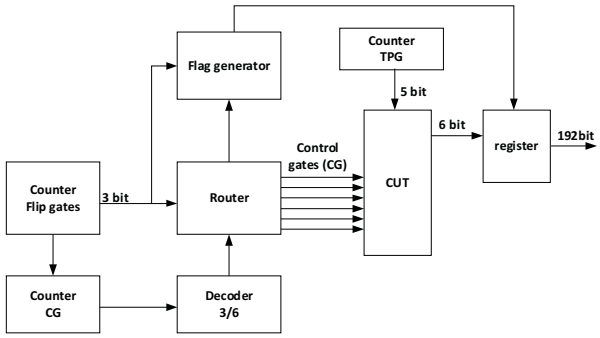



Fig. 3. Signature generation stage.

The second phase in the tool is the data processing stage (shown in Fig. 4). In this stage, the resulting signatures of the signature generation stage will be stored in the FIFO1 to be demultiplexed as (8 bits) to the FIFO2 which read the data slower and then arranged for sending through the UART module of the FPGA board as 1 bit at a time using the serial port of the computer.

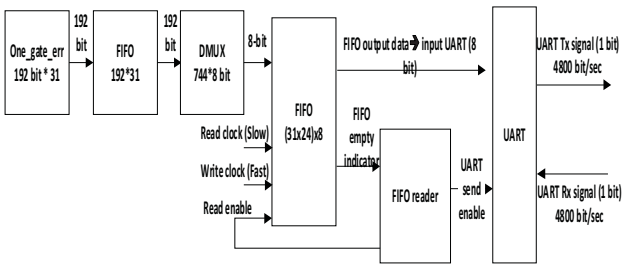



Fig. 4. Data processing stage.

The hardware testing setup was developed as shown in Fig. 5. The design will be downloaded to the Cyclone II EP2C35F672C6 FPGA in the DE2 Board (shown in Fig. 6) through UART module. The extracted data will be sent to the computer through RS232 from the board to be imported in a MATLAB software for getting the signatures of all nodes in different cases. By getting all faulty signatures that can be compared to the golden one, we can assign the faulty nodes in the CUT and then trace the CUT backward to get the source of the fault.

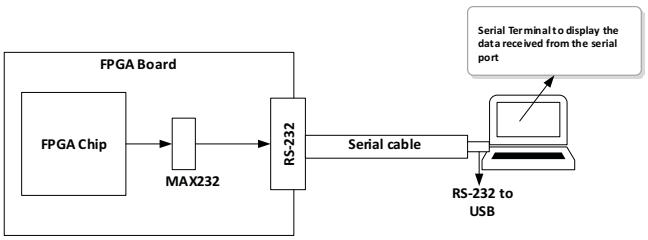



Fig. 5. Hardware testing setup.

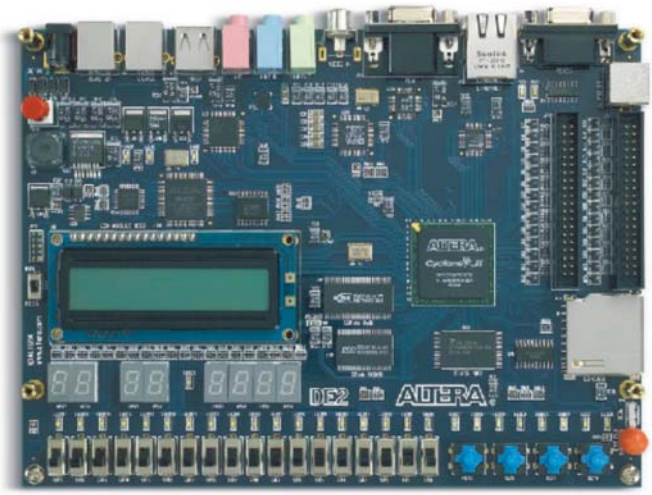


Fig. 6. The DE2 board (EP2C35F672C6 chip).

## IV. Experimental Results

Our tool produces the signatures of internal nodes in response to the exhaustive test set. There are one fault-free signature and 30 signatures for MIGSEs from the circuit of which 4 are listed in Table 1.

Table 1. Signature of C17 nodes.

| Node | Original signature | CUT signature1 | CUT signature2 | CUT signature3 | CUT signature4 |
|---|---|---|---|---|---|
| N10 | DD33EF23 | DD33EF23 | DD33EF23 | DD33EF23 | DD33EF23 |
| N16 | DCFEDCFE | *98BA98BA* | *32323232* | *98BA98BA* | *76767676* |
| N11 | 77EF9D8D | 77EF9D8D | 77EF9D8D | 77EF9D8D | 77EF9D8D |
| N19 | 62EA62EA | *FBFBFBFB* | *FBFBFBFB* | *EAEAEAEA* | *62EA62EA* |
| N22 | CCBC7434 | CCBC7434 | CCBC7434 | CCBC7434 | CCBC7434 |
| N23 | EEFEEEFE | *CCFCCCFC* | *99B999B9* | *6E7F6E7F* | *BBBBBBBB* |

The nodes that have faults in their signatures are distinguished (shown in italics in Table 1) so by tracing back from the output to locate the source of the faulty node.

It is found that the nodes N16, N19, N23 have faults in their signatures, so by tracing back from the output to locate the source of the faulty node, we assign the faulty node of the error as shown in Fig. 7.

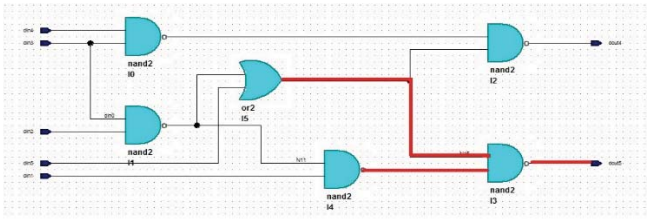



Fig. 7. Error location in c17.

It is found that time executed during emulation phase in the FPGA is 19.840µs by using 50 MHz for the UART module. The time taken in MATLAB for getting the output signature is 0.461045 s, so the total time is 0.461065 s. A detailed summary produced by the hardware synthesis software is shown below.

```
Flow summary:
Family  Cyclone II
Device  EP2C35F672C6
Timing Models  Final
Total logic elements   659 / 33,216 ( 2 % )
Total combinational functions 419 / 33,216 ( 1 % )
Dedicated logic registers     576 / 33,216 ( 2 % )
Total registers        576
Total pins     10 / 475 ( 2 % )
Total memory bits      204,800 / 483,840 ( 42 % )
Embedded Multiplier 9-bit elements  0 / 70 ( 0 % )
Total PLLs     1 / 4 ( 25 % )
Total logic elements   683
Total combinational functions 417
Dedicated logic registers     576
Total registers        576
Total pins     10
Total memory bits      204,800
Embedded Multiplier 9-bit elements    0
Total PLLs     1
```

We further experimented with another combinational circuit (shown in Fig. 8) with 4 inputs and 32 outputs to analyze the effect of the signature analysis. We built a test bench that simulate the MIGSE error model and getting the signature of each node of the circuit after replacing each gate according to the selected control gate from the library.

The output signatures of the CUT (512bit) will be stored in the FIFO1 to be demultiplexed as (8 bits) to the FIFO2 which read the data slower and then arranged for sending through the UART module of the FPGA board as 1 bit at a time using the serial port of the computer.

The simulation waveforms that demonstrate the flipping of gates during the error simulation are indicated as shown in Fig. 9.

The simulation waveforms of the output signatures of the UART module are indicated as shown in Fig. 10.

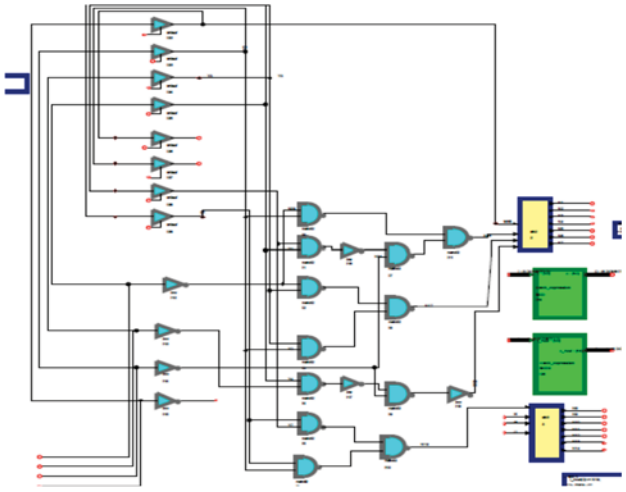


Fig. 8. The second example combinational circuit.

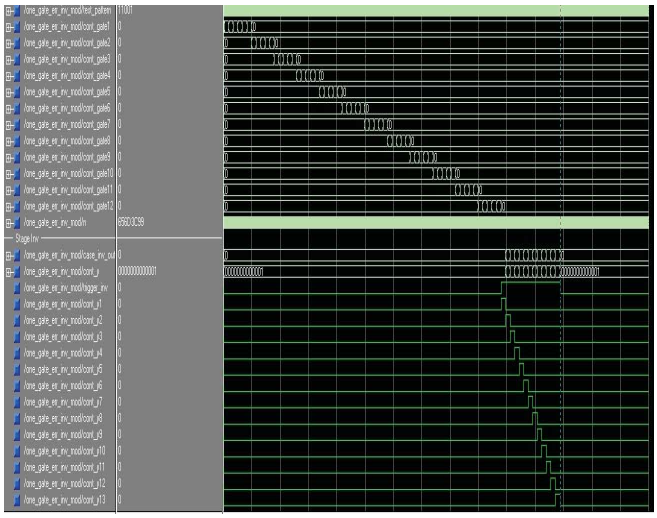


Fig. 9. The combinational circuit wafeforms during error simulation.

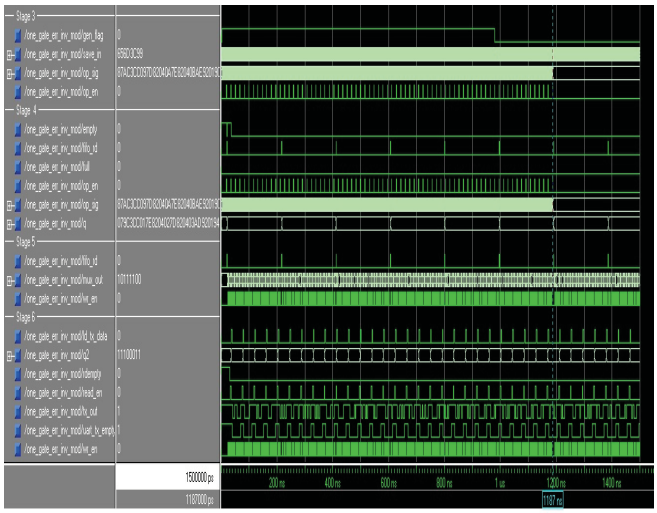


Fig. 10. The combinational circuit wafeforms of output signatures.

By tracing back from the outputs of the circuit to locate the source of the faulty node, we can assign the source of the faulty node as shown in Fig. 11.

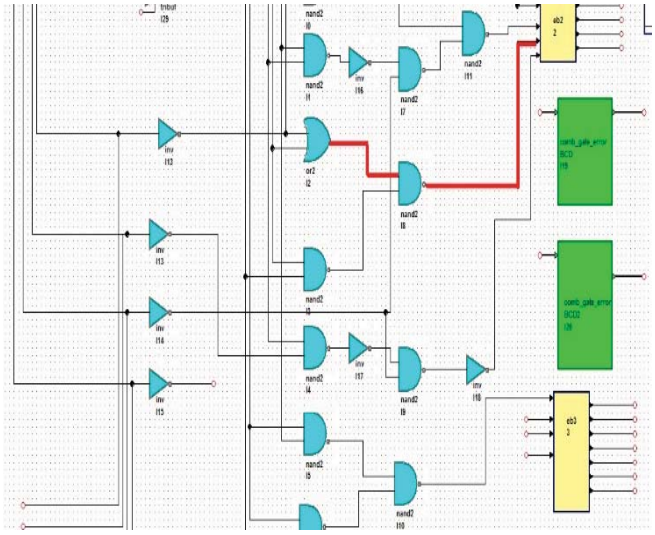


Fig. 11. The combinational circuit error location.

It is found that the time executed during emulation phase in the FPGA is 23.68µs by using 50 MHz for the UART module.

Our experimental results show that the exhaustive test set can cover 100% fault detection & localization of all gate errors for the desired fault simulation model.

## V. CONCLUSIONS

The proposed system generates and transmits the test patterns to the CUT inside the FPGA-based board, computes the response and generates the signatures on the board that are later transmitted back to the computer to be used later for fault detection and location.

The paper produced a complete framework for the testing of digital integrated circuits by converting the VHDL code of the design into an FPGA implementation that may be considered as a portable ATE.

In the proposed approach, direct interfacing between the CUT (combinational circuits) and the computer, as well as accuracy and speed of operation are desired. The speed of operation is achieved by generating the signatures of all nodes simultaneously. The system achieves the following main goals: More accurate diagnostics, great fault converge, reduced time to test and diagnose, minimum program preparation time, standardized system, reduced hardware cost, reduced operator skill requirements, and improved operator interface.

Our future work can incorporate more types of fault models into the package. We can also concentrate on applying deterministic or pseudorandom test pattern generations since exhaustive tests are not possible for large circuits.

## REFERENCES


[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, New York, 1990.

[2] M. S. Abadir, J. Ferguson, and T. E. Kirkland, "Logic design verification via test generation", *IEEE Transactions on Computer-Aided Design*, Vol. 7, pp. 138-148, January 1988.

[3] S. Kang and S. A. Szygenda, "The simulation automation system (SAS); Concepts, implementation, and results", *IEEE Transactions on VLSI Systems*, Vol 2, No. 1, pp. 89- 99, March 1994.

[4] S.-Y. Huang et al., "ErrorTracer: A fault simulation-based approach to design error diagnosis", *Proc. International Test Conference*, 1997, pp. 974-981.

[5] D. Nayak and D. M. H. Walker, "Simulation-based design error diagnosis and correction in combinational digital circuits", *Proc. VLSI Test Symposium*, 1999, pp. 70-78

[6] H. Al-Asaad and J. P. Hayes, "Design verification via simulation and automatic test pattern generation", *Proc. International Conference on Computer-Aided Design*, 1995, pp. 174-180.

[7] E. J. Aas, T. Steen, and K. Klingsheim, "Quantifying design quality through design experiments", *IEEE Design and Test*, Vol. 11, No. 1, pp. 27-37, Spring 1994.

[8] M. El Said Gohniemy, S. Fadel Bahgat, Mohamed H. El-Mahlawy, and E. E. M. Zouelfoukkar, "A novel microcomputer based digital automatic testing equipment using signature analysis," *Proc. IEEE conference on Industrial Applications in Power Systems, Computer Science and Telecommunications*, 1996, pp. 13-16.

[9] H. Al-Asaad and J. P. Hayes, "ESIM: A multimodel design error and fault simulator for logic circuits", *Proc.IEEE VLSI Test Symposium* , 2000, pp. 221-228.

# Integrated Graphical Program in Lumousoft Visual Programming Language

Xianliang Lu
Lumousoft Inc.
Waterloo Ontario Canada
lu.x@lumousoft.com

**Abstract** - ***This paper presents integrated graphical program in the Lumousoft visual programming language. Graphical programming language can be layout in two dimensions, consequently, it can be connected in all directions with other graphical programs in a plane, not like textual language which has only two directions, entry and exit. As we know, in the electric industry integrated circuits (IC) are widely implemented to represent a complicated circuit that brings in great benefits in our daily life. Based on the idea of the integrated circuit, we can use a block to represent a complicated graphical program with multiple output and input sockets to facilitate software design, implementation and maintenance. This kind of block is defined as an integrated graphical program (IGP). In this paper the integrated graphical program (IGP) is discussed in detail as well as interference sockets.***

***Key words - integrated graphical program, layout, Visual programming Language, 2 dimensions, socket***

## 1 Introduction

Lumousoft visual programming language allows software program to be presented in the form of schematics with blocks connected in parallel and series, similar to the electrical circuits which represents the electrical current flow in parallel or series circuits. When we look at an electric circuit, a part of electric circuits can be subtracted and replaced with a block named as integrated circuits. The block represents the subtracted part of electric circuits to perform special functions or task. The integrated circuit has multiple connectors or pins with the other part of circuits, and exchange information with the outside world. The integrated circuit not only enable the electronic designer to design more reliable and cost effective products, but also play an important role in our modern life.

In textual programming language a function represents a collection of program to perform special functions, limited by one dimension, the function has only one input and one output or return to make sequence flow from input to exit. As for some visual programming languages like Labview, Simulink, the pre-built modular blocks are employed to be connected together to make a program. Each modular block may have multiple input and output terminals like integrated circuit (IC). A modular block usually is treated as a parallel process in the program, and uses an event-trigger method to handle input signals and output responding signals. These modular blocks are generally provided by software provider, the user is very difficult to build their specific modules.

Graphical language offers more than one dimension than textual language, the graphical element in the graphical language should be able to handle all directions in a 2-dimensional plane, other than one dimension like textual language. It is well known that the more dimension it is, the more freedom. Lumousoft visual language takes the advantage of special compiling technology that allows compiling in network other than parse tree. Lumousoft graphical language allows the user to draw more complicated layout to meet design requirement with less effort.

Lumousoft visual programming language enables the executable block to be executed in sequence in the same way as a textual programming language does, but the flowing direction of the program depends on how the blocks are connected. Since Lumousoft VPL is a 2-dimensional layout language, we can subtract one part of the graphical program to construct an integrated graphical program with multiple input and output terminals for special functions just like an electric circuit. The integrated graphical program can be used as a function repeatedly, and it has multiple inputs and outputs with more flexibility and readability. Furthermore, the user can build their special IGP as simple as writing a textual function.

Therefore, in Lumousoft visual language Integrated graphical program (IGP) can be one of programming approaches that allows the user to have more choice to design his or her software program.

## 2 Call Graph

Call Graph is similar to the function in a textual language. The function can be broken down into two classes, inline and regular functions. When a program calls an Inline function, the compiler just duplicates the inline function body and insert where it is called. On the other hand, When the program calls a regular function, the program jump to the function routine, after executing the function, it will return back to where it is called. Obviously, a function looks like 2-pin integrated circuit, one pin for input, the other for output. Because the Integrated Circuit (IC) is independent in the circuit, the inline function is more like an integrated circuit.

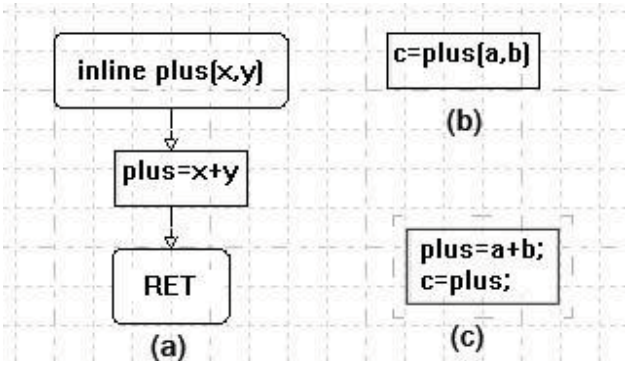



**Fig. 1 inline call graph and euqivalent graph**

Lumousoft visual programming language has an inline call graph which is the same as the textual inline function. When an inline call graph is called in an application program, the compiler will duplicate the call graph body and insert it into where it is called. Furthermore, all the variables or parameters in the duplicated graph, that declared in the call graph are replaced by the variable or arguments that passed by the graphical program who calls this inline graph. Fig.1 illustrate this duplication, the fig.1 (c) is the equivalent graphical program. Therefor we can consider the inline call graph has one entry and one exit, and can be easily inserted into one branch or path of program.

# 3 Integrated Graph Structure

A programming element with one entry and one exit can be easily put into one way flow path, especially for one dimension and parse-tree based programming language. As for 2-d graphical programming language, we can extend the inline call graph to multiple input and output element that is connected in a network.

Fig.2 is the Lumousoft graphical code, we can cut out the part which is enclosed by a dot-line block and replace it with the integrated graphical program (IGP). Consequently, there are five connection terminals, two inputs and three outputs for a cutting-off part or integrated graphical program(IGP). Since the connection line contains flowing data, we need to decide which variables need to be passed to the integrated graphical program (IGP) . Therefore, we need to make socket for each terminal. For simplicity, we can use the same format as a function. The format for the socket is:

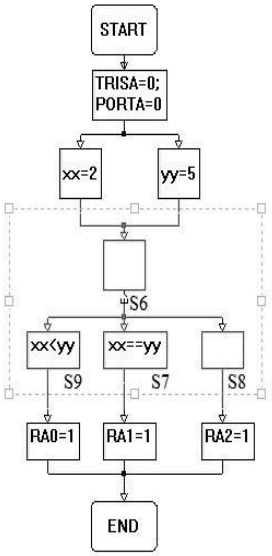



**Fig. 2 gaphic program layout**

socket_name (parameter1, parameter2,…)

Where socket_name is the name of the socket.

In Lumousoft language the function name is a variable, so is the name of a socket, usually the data type of socket name is defined as void; parameter1, parameter2 are the parameters that want to be replaced in IGP. In the input socket, the parameters can be passed into IGP through arguments or variable of application program. While in the output socket, the parameter can be passed out to the application program.

Fig.3 illustrates that the IGP has two input and three output terminals. The input terminal is represented by the filled arrow symbol, the filled square bar stands for output terminal. The IGP symbol is represented by a block with a bottom left triangle, the context of the IGP block shows the name of IGP and socket as well as its arguments that are passed to the IGP graphical program. On the left hand side of the Fig.3 is the

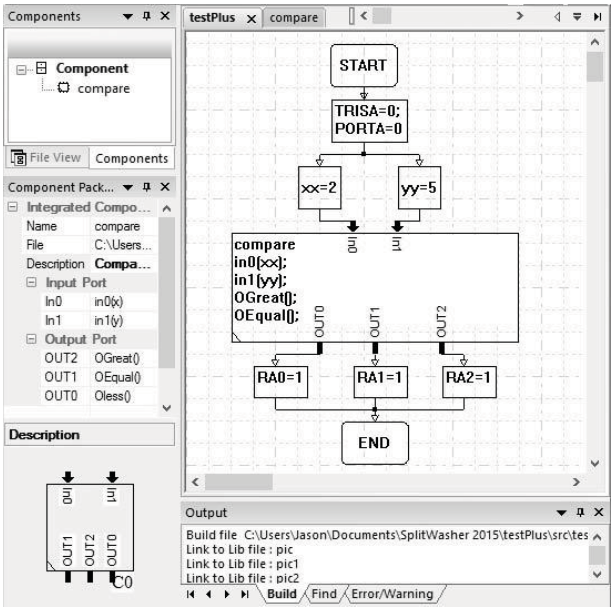



**Fig. 3 Graphic program with IGP**

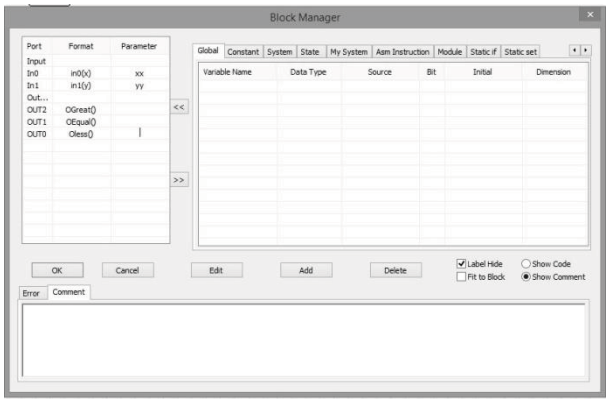



**Fig. 4 Block manager diaglam**

pane of IGP property and its image as well as component list tree in which the components are available for application

To setup socket parameters, just double click the IGP block In Lumousoft Visual Programming Language IDE, a block manager diagram will pop up as shown in Fig.4, fill the corresponding variable in the cell of parameter on the left pane. Close the block manager diagram, all the sockets will automatically be displayed in the IGP block as shown in Fig.3. The connection terminal symbol can be relocated in position. Unlike the regular symbol on the normal block, the terminal symbol of IGP cannot be deleted, added or change direction in implementing program.

Integrated Graph Program acts as a black box, how the data flow over IGP depends on the inside program path of IGP. If the output socket has one or more paths connected to one input socket, the living variable, which is alive after the IGP, will go through this input socket and pass out through this connected output socket. The inside local variable in the IGP cannot pass out to the application program, except that the local variable is socket parameter.

Input socket is responsible for passing application program arguments into IGP. While the output socket gives the door to exit IGP,and pass the result of IGP out to the outside program.

# 4 Integrated Graphical Program (IGP)

## 4.1 Composition

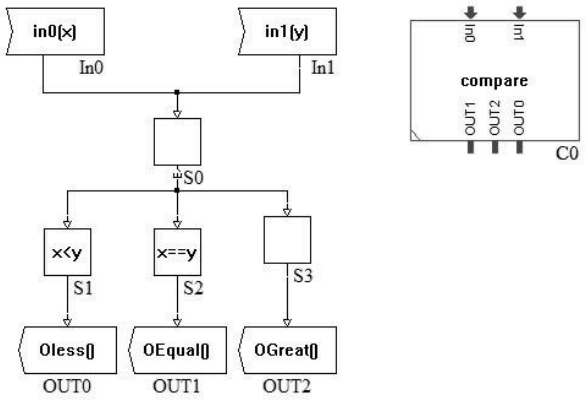



**Fig. 5 Setup IGP component**

It would not take more efforts to build an Integrated Graphical Program (IGP) than to write a textual function code. Lumousoft VPL IDE offers an IGP building tool to accomplish IGP component. As we mentioned before, we need input and output sockets for interference with the outside program. The input socket symbol is a rectangle with a triangle cutting off on the left hand side as the symbol with label In0 shown in Fig.5. The output socket symbol is a rectangle with an extra triangle on the left hand side as the symbol with label Out0 shown in Fig.5. The input or output socket name and parameter can be edited through the block manager diagram in Lumousoft Visual Programming Language IDE. When setup a new IGP component, an IGP block will be automatically displayed on the upper right corner of the screen. Whenever place a new input or output port socket, a new corresponding connection terminal symbol or pin (filled arrow or square bar) will automatically be attached to the IGP Block; If one socket port is deleted the corresponding pin in IGP block will be deleted too. Therefore, the outlook of IGP component can be seen simultaneously during building. The body of graphical program can be regularly layout for the special task or function.

## 4.2 Nest

An IGP component may nest other components to make a multiple hierarchy level component. Fig.6 illustrates the nested component. Because Lumousft VSL is designed for microprocessor, and the microprocessor has limited memory

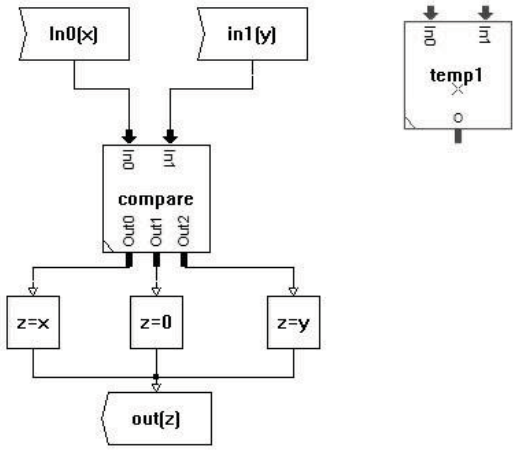


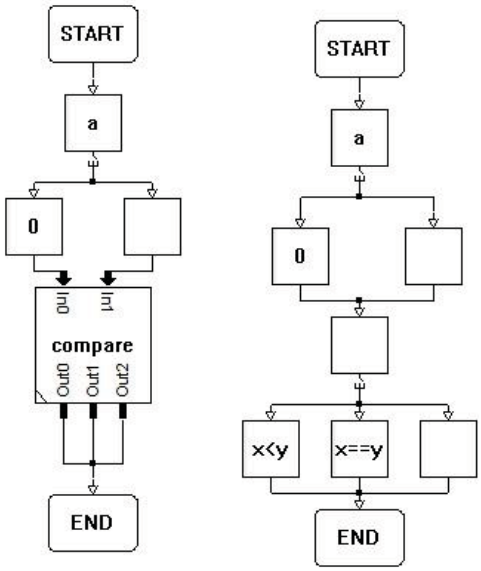

**Fig. 6 Nest component**

resource, therefore, it cannot employ recurse technology. In the other words, the deeper nested components cannot contain the high level nested component.The Lumousoft VSL compiler can find this kind of recurring errors.

In Lumousoft graphical language, the IGP must be built before its implementation. The inside IGP will be replaced by the corresponding IGP body after building.

## 4.3 Syntax Error Check

When an IGP component is accomplished, the inside paths need to be verified whether it violates the path rules that allow the compiler successfully compile and make graphical program correctly and logically. As we know a component which has multiple input and output sockets, it cannot run itself. In order to verify it, the compiler automatically puts the new component into a simple graphical routine with a start and end blocks as shown in Fig 7 (a), the Fig 7 (b) is an equivalent graphical program. Each input terminal is connected to switch block, therefore the inputs are in the select branches; while all the outputs are connected to the end block. With this verification main graphical program, the new component can be checked whether the inside paths in the new component has syntax errors and warnings.




(a) Tset program (b) equivalent program

**Fig. 7 Test main program and equivalent program**

The component is like a black box, and the compiler does not know whether the inputs are designed for parallel, select or sequence input. Since select branch has not special rule, it cannot cause violation against the rules. On the other hand, if we use a parallel branch to plug into each input socket, it may cause errors, for example, if these parallel branches cannot join somewhere or these parallel branch jump out of a loop without joining together with other parallel branches. With this test program, the new component can be compiled as a regular Lumousoft graphical program. If no error is found, the IDE will automatically add the new IGP component to the component tree as shown in the upper left pane in Fig.3. The new built component is ready for user implementation.

When we want to use these components in the component tree for our program, the component can be drag and drop into the application program. Lumousoft Visual Programming Language allows the user to modify an IGP component after it is used in applications. When the IGO component is modified and updated, any component that used in user graphical program will be automatically updated to the latest one including that one in component tree .

# 5 Implementation

The above example IGP component we just built is used for comparison of two input terminal x, y. When x is lesser than y the program will go through the output terminal Out0; if x is equal to y, the output terminal Out1 will be the path for output; when x is greater than y, Out2 will be selected for output. Currently our example graphical program is designed for microprocessor PIC16F74, three port A pins are used to show the comparison result of two variables (xx, yy). These pins

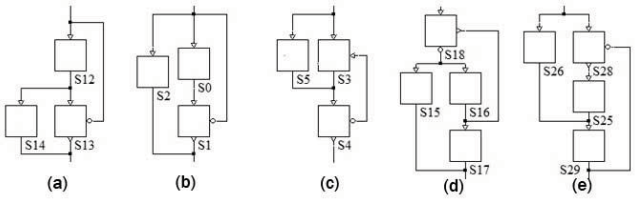



**Fig. 8 Illegal parallel connection**

which are configured as the PIC portA, RA0, RA1 and RA2 respectively correspond to the result of lesser, equal and greater. The Fig.3 demonstrates the graphical program layout for this task.

Before Compiling, the compiler first duplicate the graphical program of IGP component and insert this duplicated graphical program into where it is implemented, plugging into the corresponding socket. Secondly, all the variables in the IGP graphical program that declared in the socket will be replaced by the arguments or variables that outside graphical program wants to pass through. Finally, delete the connection terminal and make direct connections, the equivalent graphical program is shown in Fig.2. After handling all the IGP graphical circuit, the compiler will perform the compiling job, including path analysis, syntax analysis and generation of assembling language and machine code.

# 6 Caution

Inline call graph has only one entry and exit, it is always in one graphical branch or path, and it is relatively easy for parse-tree based language to handle. While, the integrated graphic program (IGP) has multiple terminal and in different network that make more complicated. When a component is put into the application program, it should be beware of its connection and understand how the GIP work, just like electric IC, otherwise it can cause program crash.

Usually the select branch like switch or true/false branch will not cause a syntax path error, select branch just like "goto" instruction technology that will not cause programming running failure. On the other hand, the parallel branch might cause syntax failure, there are restrict rules on that, Fig.8 shows the illegal parallel layout. Therefore, when we insert a component into the application program, we need to combine the application program and component program to make sure the flow path is not in illegal paths.

## 6.1 Variable Value

Because Lumousoft uses variable replacement technology, the value of a variable that is passed into the IGP may change in this black box. To avoid this change, we can assign the value of a variable to a buffer variable, and then use the buffer variable to be passed to the IGP. Fig.9 is the example of how to use buffer variable t to keep the original variable unchanged.

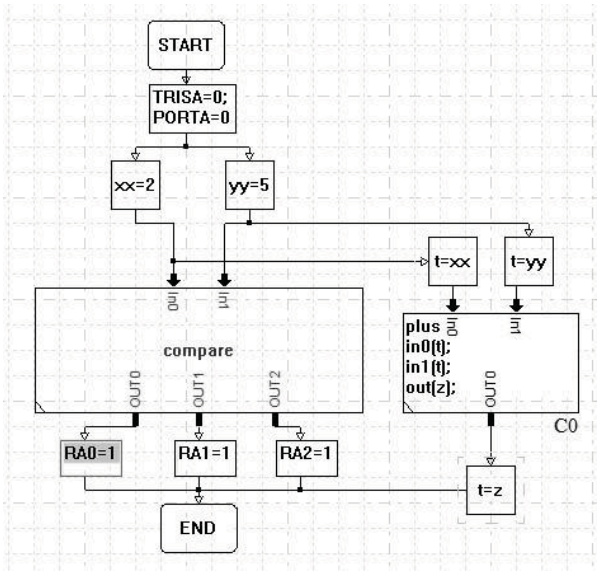



Fig. 9 buffer input for an IGP

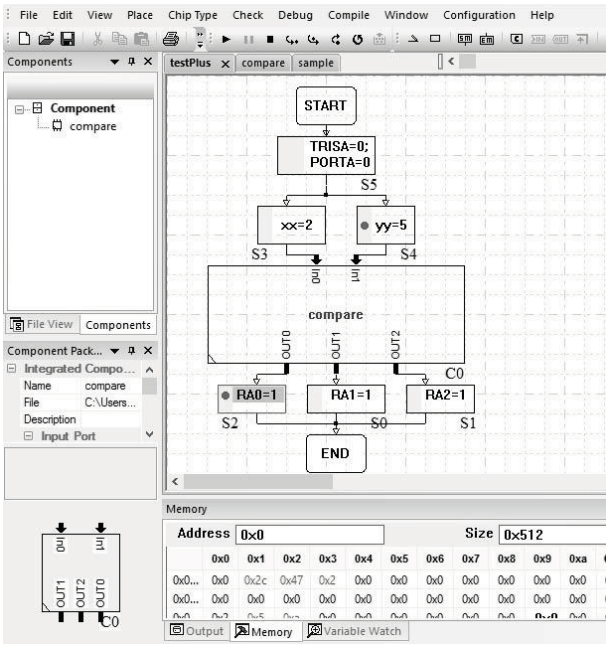



Fig. 10 graphic program during debugging

## 6.2 Variable Data Type

In Lumousoft VPL, the data type of parameter that IGP defines in a socket is meaningless, since these variables are totally replaced by the outside variable, including data type. However, some variable inside the component need to cooperate with the socket parameter, the data type of this cooperated variable must meet the data accuracy. We can apply the data type that requires maximum byte to fulfill our data accuracy, but this lead to the requirement of more hardware resource and more size of the codes that might increase hardware cost and slow down program executing.

In order to fulfill the requirement of data accuracy and optimize the program performance, Lumousoft VPL language provides some special functions to dynamically setup data type of a variable. For example, the series function _static_set(26,x,y) is used to make the data type of x to be the same as that of y. The first parameter of digital number 26 is the index number of this series function of _static_set. Therefore, the data type of some inside variable of IGP can be dynamically changed based on the user requirements.

These kind of series function will not take action until the main graphical program is compiled. In this way we can keep data type consistency. For example, the data type of a variable depends on the a parameter that its parent component will pass to. If the data type of this parent argument or variable is uncertain, in other words, the data type of this parent argument is still decided by these series functions or this parent argument will be replaced by grandparent variable. Therefore, the data type of a variable cannot be decided until final compilation

When building component or inline function, if the compiler come across the _static_set series function, the compiler will skip. However, when final compiling, the compiler will give each variable a definite data type and interpret these _static_set series functions.

## 6.3 Same Variable Parameter in Different Socket

Usually, it needs to avoid defining the same variable in different sockets for an IGP component, the compiler will not consider it is an error, but it may cause failure. Because the different outside variables want to replace the same parameter of IGP, after compiling it will be found that only one outside variable replaces this parameter, the others not. To avoid this, we can use a buffer variable for outside variables to assign their value to it, and then use this buffer variable to pass to IGP. Fig.9 demonstrate the buffer variable application for component C0. The two input socket has one same parameter x, if we directly pass xx to the parameter x in the socket In0 and yy to the parameter x in the socket In1. When x is replaced by xx first, because all the x is replaced by xx, there is no x to be replaced by yy. That might cause bugs. The solution is to use buffer variable to avoid this situation.

# 7 Debug and Test

The textual language IDE usually offers debug tool to trace program running to ensure software quality. Lumousfot also offer this debug tool and be able to trace program flowing.

Lumousoft Visual Programming Language IDE can be offline to simulate microprocessor for debug. Similar to the regular textual language IDE, Lumousoft IDE allows the user to implement debugging methods like Step into, Step Over, Step Out, Run, Stop, Breakpoint, Reset, etc. for offline debug. Watch window can be used to observe the value of the variable during debugging. Fig.10 shows the graphical program during debug and the program stop at the block S2 with a highlight bar, which can follow the program flow and indicate where the program run to. Currently, it stops at Block S2 as shown in Fig.10, the output terminal OUT0 is the only output when the value of xx is lesser than yy. The location that program stop in

block S2 is what we expected, indicating that the graphical program is correct. The filled circle on the left hand side of a block,such as in block S2, stands for the breakpoint. The program will stop when the program hit this breakpoint.

# 8 Conclusion

Lumousoft visual programming language makes a great effort to make Integrated Graphical Program (IGP) module available to the user in the similar way of the Integrated Circuit (IC) in the electronic industry field which revolutionize our modern life. The programmers are able to build their Integrated Circuit Program(IGP) modules in the same way of regular graphical programming method and make input or output port socket, in the similar way to construct a function. The programmers can use their familiar way to build an IGP component with less effort and easily implement them in their application.

Integrated Graphical Program (IGP) is like a textual inline function. When it is implemented, it just duplicated the IGP program body and insert where it is employed. By means of socket, the data flow can be kept consistent and smooth.

Lumousoft Visual Programming Language allows the programmers have more programming strategies and make their graphical program more readable and easier maintenance.

# 9 Reference

[1] Xianliang Lu, “Lumousoft Visual Programming Language and its IDE”, The 2014 International Conference on Embedded Systems & Applications, pages 3-9, July, 2014.

[2] B. Jost , Schloss Birlinghoven, Fraunhofer IAIS, St. Augustin, Germany, M. Ketterl, R. Budde, T. Leimbach,” Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One? ”, Multimedia (ISM), 2014 IEEE International Symposium on,2014

[3] WESLEY M. JOHNSTON, J. R. PAUL HANNA, AND RICHARD J. MILLAR, “Advances in Dataflow Programming Languages”,ACM Computing Surveys, Vol. 36, No. 1, pages 1–34,March 2004,

[4] Xianliang Lu, “Memory Management and Optimization in Lumousoft Visual Programming Language”, The 2015 International Conference on Embedded Systems & Applications, pages 10-16, July, 2015.

# Robotic Ball Collection System

Bassam Shaer, Joshua Keeman, Lucas O'Neill, Jonathan Rogers
Department of Engineering, University of West Florida, Fort Walton Beach, FL
bshaer@uwf.edu, jpk23@students.uwf.edu, ljo4@students.uwf.edu, jlr82@students.uwf.edu

**Abstract—** *Golf driving ranges can amass 35,000 golf balls a week. This requires cleanup by an employee while the range is active. This process requires a lot of time and money due to current machines and processes. This paper contains a proof-of-concept solution which involves the design, construction, and demonstration of an autonomous robotic system, the purpose of which is to identify and retrieve a predetermined object in a cost efficient manner. The process of identifying this object will be implemented using image processing, and the act of retrieving the object will be performed by a one-degree-of-freedom manipulator arm. This proof-of-concept would also be well suited to a military application.*

**Keywords—** Proof-of-concept, autonomous, image processing, one-degree-of-freedom

## *I. Introduction*

On any given day, there is a large number of golf balls that are hit onto an equally large area of land set aside for golfers to practice their golf swing. These golf balls must be retrieved for reuse and the process of collecting them can be time-consuming. In most cases, this task is performed by an employee of the golf course/country club, sometimes during a period of high activity on the driving range.

The goal of this paper is to explain the way to automate this process. Using an automated system for this purpose would improve the safety of employees and reduce long-term costs for an employer in terms of man power and resources. A large scale device that can do this is not achievable at this stage but one has been created at a smaller scale. In this case, marbles. Instead of a golf course, portions of a basketball gym are being used in an organized fashion. The device is programed to use the processor to search for marbles and send a message to the drive control function to go after a marble if any are detected. The intended users of this concept are personnel of golf courses/country clubs.

Automation continues to play an increasing role in lives of many people more and more each day. One of the biggest benefits of automation is that it saves labor and its costs along with saving energy and materials [2]. Automation is also more accurate than humans and the quality of production improves anywhere automation is used, like in mass production. Automation has been an achievement in many fields such as electrical, mechanical, and hydraulics. This type of automation is used in complicated systems such as military equipment, airplanes, and factories.

Two processors were used in this automation process: a Raspberry Pi 2 model B and an Arduino. The Raspberry Pi 2 consists of 1 GB RAM along with 40 GPIO pins, Camera interface (CSI), and a Display interface (DSI) in order to control the camera and receive signals from the camera confirming that a marble is present [7]. The Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online [6]. One can tell the board what to do by sending a set of instructions to the microcontroller on the board. To do so one uses the Arduino programming language (based on wiring), and the Arduino Software (IDE), based on Processing [3, 4].

The motors used for this device can only handle around 5 pounds and have to be used for a limited time so that they do not burn out whenever the weight limitation is approached. The Raspberry Pi does not have processor as high as a computer. Therefore, the camera attached to the Pi has a frame rate of about 2 fps. The robot cannot move too fast for these reasons.

The design runs over the full area of the range, as well as the distance required for it to return to the base/charging station. The drive system and system batteries are rugged enough and of high enough capacity to handle this task. As a design of this scope is costly and a time-consuming venture, that is, outside of the realm of possibility of the design group's time and budget constraints, it was decided to scale down considerably the scope of the design to a unit that collects marbles off of a section of a gym floor, within a predetermined area.

This design is better than other designs similar to this because this robot is completely automated based on commands provided by the user into the programming. For example, a user can input what color marble he or she desires and the robot will detect that colored marble and retrieve it in an efficient manner. This can also be used as a proof-of-concept for military applications such as mine detection in order to protect the lives of those serving in the military. Most devices similar to this one either simply follow an object or are manually controlled by the user. The rest of the paper is organized as follows. A brief review of the functional block has been discussed in section 2. Section 3 presents an overview of the hardware. In section 4 the formal description of the proposed software and its operations has been presented. Section 5 presents the testing approach. The paper ends with concluding remarks in section 6.

## II. Functional Block Diagram

Below is the functional block diagram with brief descriptions of each function in order for the robot to search, find, and retrieve marbles in an efficient manner.

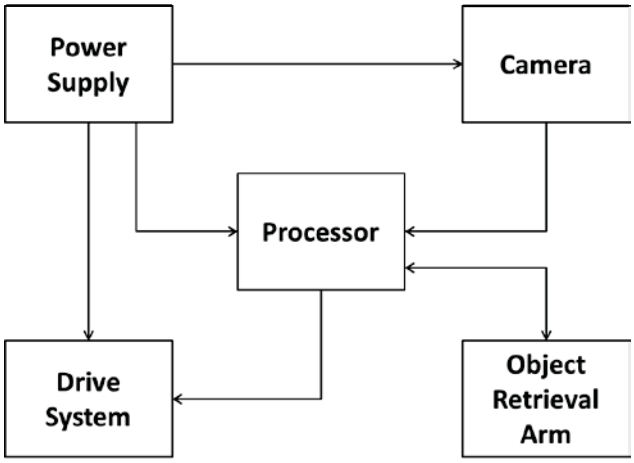



**Figure 1.** Functional Block Diagram

The following are brief descriptions of the desired functions for each block in the block diagram:

- The processor block is used for sending, receiving, and processing signals. It controls all system functions.
- The power supply block provides power to all system components.
- The drive system block contains the motors and wheels that are necessary for the mobility of the robot.
- The camera block is used to detect the marbles and send the signal to the processor.
- The object retrieval arm is commanded to pick up the marble and place in the storage container.

## III. Hardware

The hardware for this robot is designed in such a way so that the robot can search, find, and retrieve marbles in an efficient manner. The functions for each piece of hardware are described below.

### a. *Power Supply*

The circuit schematic for the power supply is displayed in Figure 2 below while the finished product is shown in Figure 3. The power supply receives an 11.1V signal from a lithium ion, rechargeable battery and utilizes two L7809 voltage regulators producing two 9V signals and three L7805 voltage regulators producing three 5V signals. The two 9V regulators are connected to the Motors and the Arduino with currents at 1.5A each. Two 5V regulators are used in connection with the Raspberry Pi because the Pi requires at least 1.8A in order to power up and function properly. The third 5V regulator provides power to all of the servos at 1.5A. There is a manual override by means of a power switch should there be issues. Once activated, the power shuts down on all of the components. A rechargeable battery is connected so that whenever one runs out, another can replace it. On the final power supply, the red wires represent the positive voltage while the black wires are connected to ground. Heat sinks were included in order to prevent the voltage regulators from overheating.

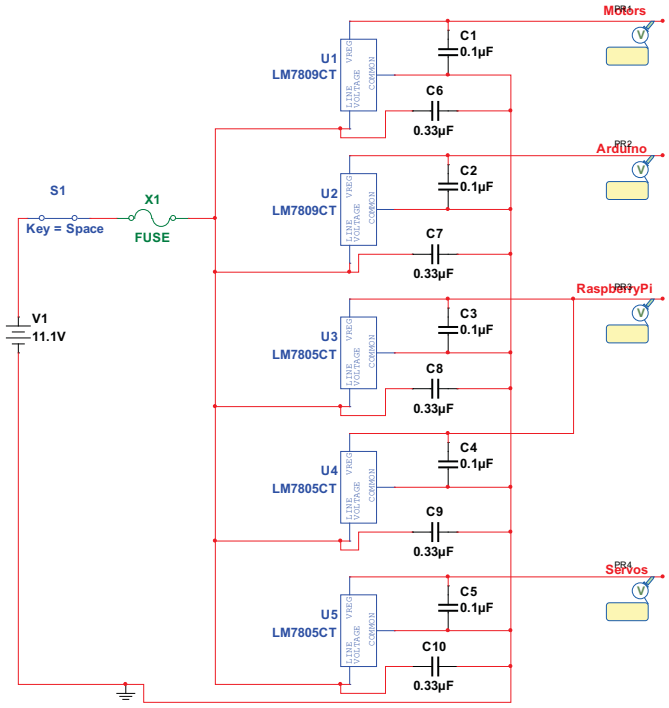



**Figure 2.** Multisim Circuit schematic for the power supply.

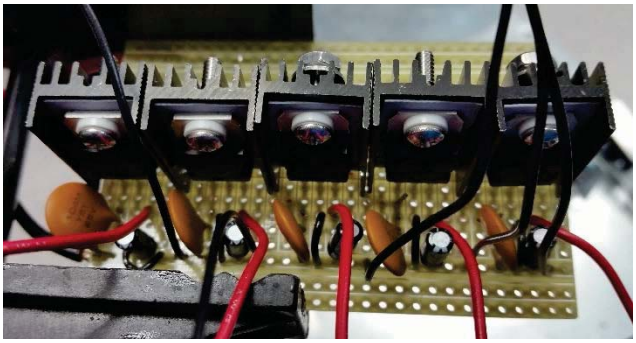


**Figure 3.** The Power Supply.

### b. *Drive System*

The drive system consists of all robotic functions involving the movement of the robot itself. This system contains the Arduino connected to a dual H-bridge. The dual H-bridge separates the signals to each of the two motors in order for the robot to change direction. Figure 4 displays the dual H-bridge with motors attached. The flowchart in Figure 6 below contains the functions of the signal drive system.

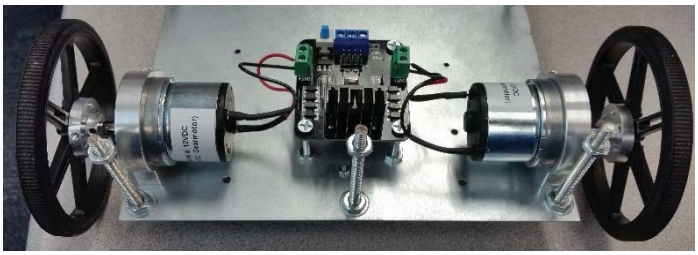


**Figure 4.** The H-bridge with motors attached.

*c. Processor*

The processor consists of the Raspberry Pi. This sub-unit is used for the processing and control of all system functions throughout the robot. The Pi receives wireless control signals and sends wired control signals to other system components such as the Arduino and camera. The Pi also detects signals for system start and stop and receives feedback from the drive system motors and the camera. The Pi communicates with the Arduino by sending signals to activate and to deactivate certain robotic functions. The Pi has a manual system override by means of ending the program from a wireless keyboard. The Pi also powers up the camera in order for it to search for marbles. Section g, Processor *Pin Connections,* below goes into greater detail on what pins were used for each device and how they are connected. Figure 10, contains the programming process for the image processing functions performed by the camera and Raspberry Pi.

*d. Camera*

The Camera detects marbles in a set area and centers itself with the closest marble as shown in Figures 8 and 9. The camera scans left and right in order to find a marble while using pattern recognition to find a circle and send the signal to the processor. The image processing functions are explained in greater detail in the section called "*Image Processing"* below. The camera is placed on the front end of the robot and is hardwired to the Raspberry Pi. The camera receives power from the Raspberry Pi and provides feedback signals to the Arduino in order to activate the robotic functions. The camera resolution is 800x600 pixels at a rate of 2 frames per second. A low resolution is used in order to save memory and power consumption from the power supply. This lower resolution may seem small but it is just as effective as a higher resolution in this specific case. The frame rate is at 2 fps due to the programming and is all that is necessary for this task.

*e. Object Retrieval Arm*

The object retrieval arm is used to pick up marbles anytime it is command by the Arduino based on signals from the camera. This device uses two servos to move back towards the storage container and forward towards the marble. Only one other servo is used for opening and closing the claw that picks up marbles. These servos are connected directly to the processor and communicate through the Arduino in order to move in the direction commanded by the Arduino. The retrieval arm is powered by a 5V signal from a voltage regulator instead of the Arduino due to the fact that the Arduino does not output the proper 1.5A signal in order to simply power the servos. Figure 5 below displays the Robotic arm before the second servo was included. The second servo mirrors the first servo. This figure also displays the marble retriever with its own servo attached.

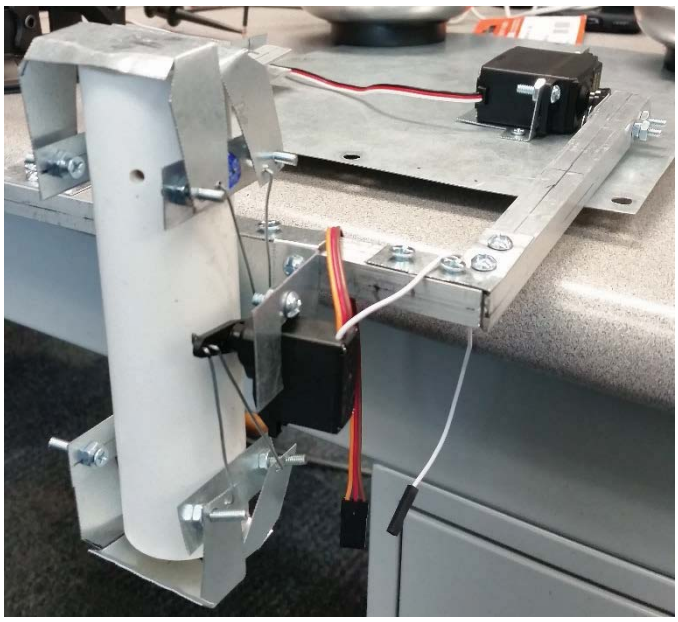


**Figure 5.** Object Retriever Arm with marble retriever.

*f. Robotic Functions Procedure*

The procedure for the signal drive functions of the flow chart shown in Figure 6 are:

1. The image processing function searches for a marble. This process repeats until either a marble is found or the camera has searched the entire area.
2. If a marble is found, the camera will send a "Marble Found" logic low signal to the Arduino. Otherwise, the Arduino will wait for a "Rotate" logic low signal to come in. If a "Rotate" signal is received, go to procedure 3. Otherwise, go to procedure 4.
3. When a "Rotate" signal is received, the robot will rotate 60° clock-wise, stop, send a "Rotate Echo" signal to the camera, and repeat the whole process. Whenever the camera receives a "Rotate Echo" logic low signal, it repeats the image processing functions by the Raspberry Pi. After the robot rotates, the Arduino will calculate whether or not this is the $6^{th}$ time the robot has rotated or not because 6 x 60° = 360°. If so, the robot program ends until told otherwise by the user.

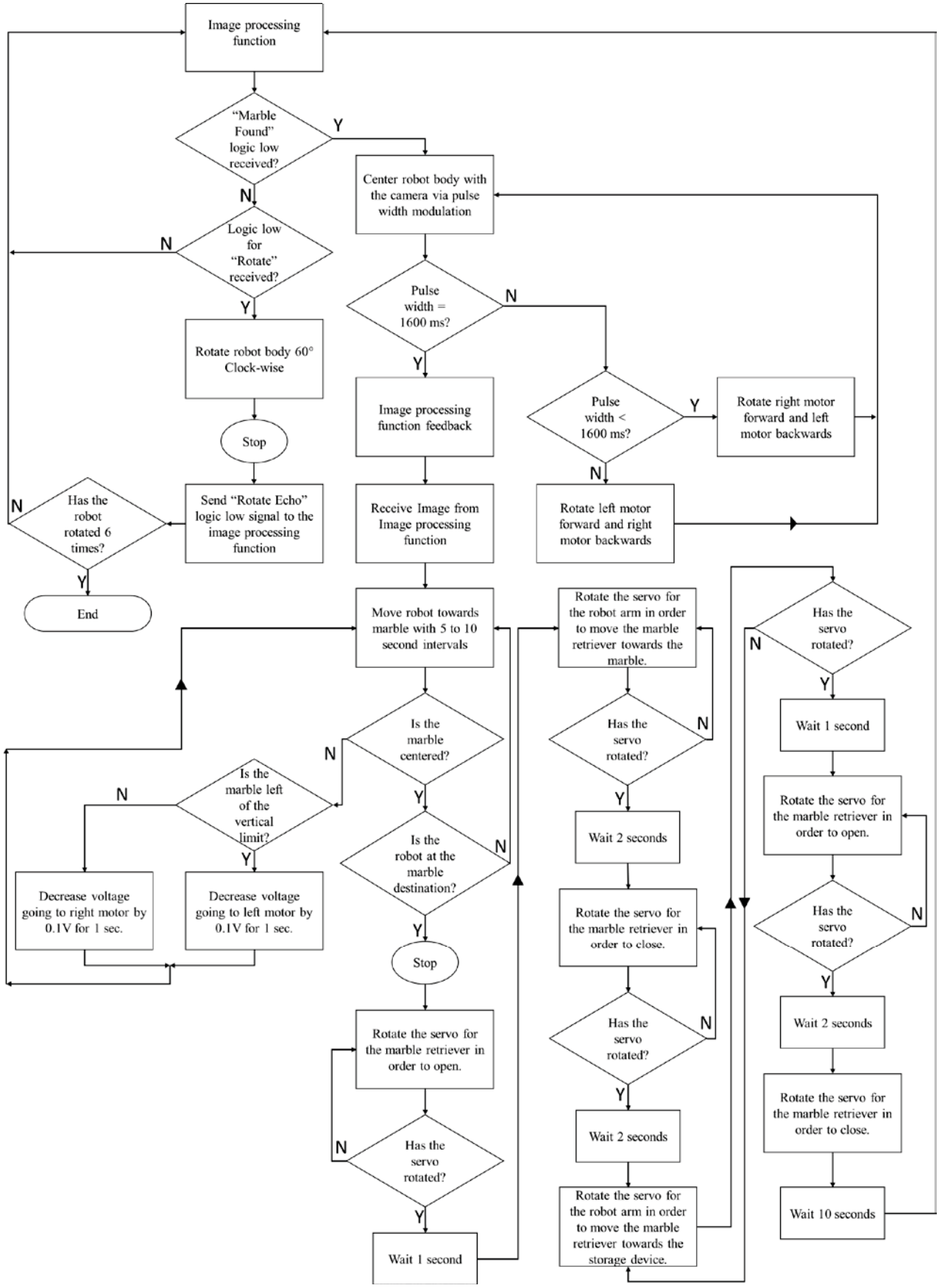



**Figure 6.** Flowchart of Signal Drive Functions.

4. Whenever a "Marble Found" logic low is received from the camera, the camera will hold still while the robot body rotates to face the direction of the marble. The robot knows when it is center based on the pulse width modulation of the camera. It is calculated that the pulse width whenever the camera is centered with the robot is 1600 ms. If the pulse width is less than 1600 ms, then the robot will rotate to the left until the pulse width is equal to 1600 ms. Otherwise, it will do the same thing but to the right.
5. Whenever the robot is centered with camera, the camera will check and make sure that the marble is still in the center of the image. If so, the pan function on the camera locks.
6. From there, the robot starts moving towards the marble. The robot will stop about every 5 to 10 seconds to check and make sure the marble is still centered. If not, the motors will rotate the wheels to line the robot back to the center based on the vertical limits provided by the camera feedback. At the same time, the camera's tilt function is operating in order to keep the marble centered based on the horizontal limits.
7. As long as the marble remains at the center of the camera's image, the robot will keep going straight. As this happens, the Arduino with the camera will constantly check to see if the robot has reached its destination. This is done through a series of calculations where the camera checks how large the marble image is.
8. If the marble reaches its destination, the claw will open, the arm will move towards the open, the claw will close around the marble, the arm will move back, then the claw will open and drop the marble in the storage container.
9. Whenever this process is done, the robot will wait 10 seconds then allow the image processing function to take over again and search for other marbles. This process will continue until the robot is told to stop by the user or whenever the robot has rotated six times without finding a marble.

*g. Processor Pin Connections*

Figure 7 displays the pin connection diagram for the Raspberry Pi and Arduino and how each device is connected to each other.

**Arduino:**

*Analog Pins*

A0 -- Tilt Feedback from Pan/Tilt Camera Mount
A1 -- Pan Feedback from Pan/Tilt Camera Mount

*Digital Pins*

2 – I1 (drive function for Motor 1)
3 – ENA (Pulse-Width Modulation for Motor 1)
4 – I2 (drive function for Motor 1)
5 – Doors Servo
6 – Arm 1 Servo
7 – Arm 2 Servo
9 – Marble Detection Feedback from Raspberry Pi
10 – Rotate Feedback from Raspberry Pi
11 – ENB (Pulse-Width Modulation for Motor 2)
12 – I3 (drive function for Motor 2)
13 – I4 (drive function for Motor 2)
19 – Rotate Echo to Raspberry Pi

**Raspberry Pi:**

Pin 12 – Horizontal Servo Control
Pin 16 – Vertical Servo Control
Pin 18 – Marble Detected signal to Arduino
Pin 22 – Rotate signal to Arduino
Pin 32 – Rotate Echo Signal input signal from Arduino

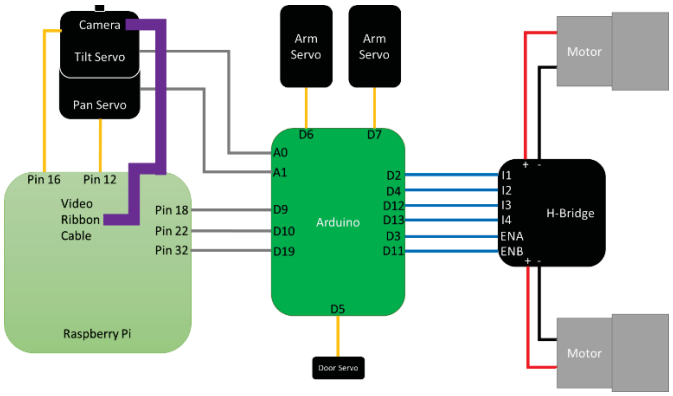



**Figure 7.** Connection diagram for each component.

## *IV. Software*

*a. Image processing Procedure*

The procedure for the image processing functions for the flow chart shown in Figure 10 are:

1. The robot is turned on and the user inputs what specific color marble he or she seeks to find.
2. The camera activates and sends the images to the Raspberry Pi.
3. If the desired object is not in the image move to procedure 4. Otherwise, move to procedure 5.
4. Whenever the camera does not immediately see a marble, the camera will rotate all the way to its left then to the right. Anytime the camera sees a marble during the search, it will go immediately to procedure 5. If the camera does not detect a marble during this process, it will activate the drive signal function with a "Rotate" logic low signal, rotate the robot 60°, and repeat the search, going back to procedure 2.
5. Whenever the camera picks up an image of a marble, the camera will pan and tilt in order to center the image of the marble in reference to the camera. This is done by checking the vertical limits first. If the marble is to the left or right of the limits, the camera will pan in that direction. After this is done, the camera will check the horizontal limits. If the marble is above or below the limits, the camera will tilt in that direction.
6. Once the camera is centered, it will send a "Marble Found" logic low signal to the Arduino in order to

activate the robotic functions causing the robot body to face the same direction as the camera.

7. Whenever the camera receives a "Rotated" signal from the Arduino, the camera locks its pan functions. When this happens, the motors will move the whole robot body left and right in order to keep the robot centered on the marble.
8. While the robot is moving, the camera's tilt function continues to operate in order to keep its "eye" on the marble and communicate with the Arduino, instructing it when to stop.
9. Whenever the robot stops, so does the image processing functions. The image processing continues only when instructed to do so.

*b. Python for image processing*

As shown in Figure 10 below, image processing is used to search and detect a marble or marbles. The way this is done is by using the programming software Python on a Raspberry Pi. As stated above, the camera searches for the marble using a camera running based on the software functions and provides feedback whenever a marble is detected. From there, the camera centers the marble image and stays there until the marble is picked up. Figure 8 displays the image received by the Raspberry Pi from the camera whenever marbles are present. The red circle shows the marble that is closest to the device. That marble is picked up first. The green circles show the other marbles in the area that were picked up later. After the camera detects the closest marble, the camera uses horizontal and vertical lines to help center the marble in the camera image as stated above. Figure 9 displays the image received by the Raspberry Pi from the camera whenever the image is centered.

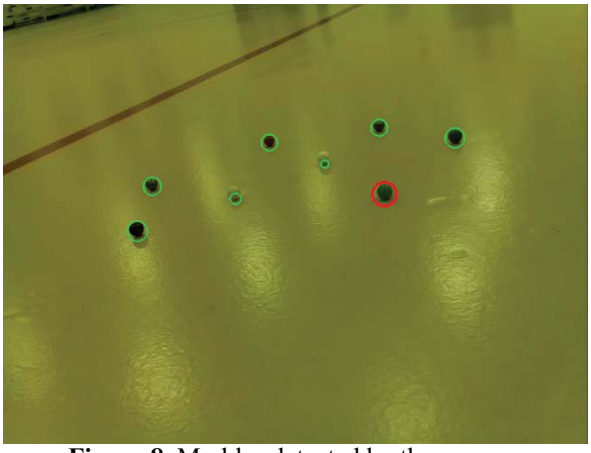


**Figure 8.** Marbles detected by the camera.

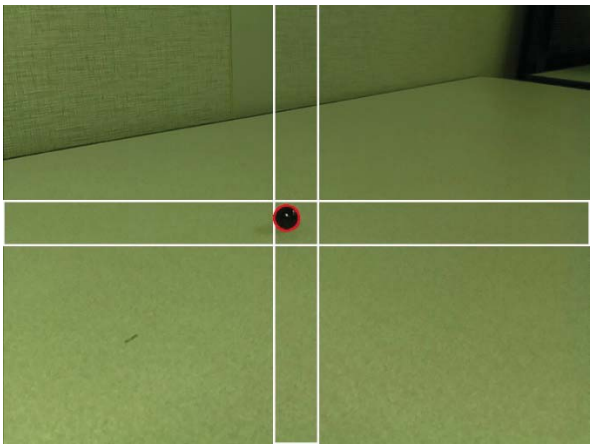


**Figure 9.** Marble centered with the camera.

## *V. Testing Approach*

In order to demonstrate that the end-product meets the design objectives and specifications, the automatic ball retrieval system is brought to the testing area, which is a gymnasium. Marbles are scattered around the predetermined area of the gym floor in a random arrangement. The robot is then placed within the vicinity of the test area to perform its functions automatically. Several arrangements of marbles and various test scenarios such as colors, location, and condition of the marbles are introduced to the robot to ensure that it performs efficiently.

## *VI. Conclusion*

The goal of this work was to design and demonstrate a fully automated system that searched, detected, and retrieved a predetermined object. This device is a proof-of-concept prototype that can be used to create an automatic golf ball retrieval system or various military devices such as land mine diffusers. While this system is not yet fully operational, it does provide the opportunity to see more clearly the steps that need to be taken in order to achieve the goal of having this device pick up marbles autonomously.

## *References*

(1) S.B. Niku, *Introduction to Robotics.* Upper Saddle River, New Jersey: Prentice Hall, 2001.

(2) M.W. Spong, S. Hutchinson, M. Vidyasagar, *Robot Modeling and Control, 1st Edition.* New York, New: John Wiley & Sons, 2004

(3) J. Warren, J. Adams, and H. Molle, *Arduino Robotics*. New York City, NY: APress, 2011.

(4) J. Blum. *Exploring Arduino*. Hoboken, NJ: John Wily & Sons, Inc., 2013.

(5) R. Grimmett, *Raspberry Pi Robotics Essentials*. Birmingham, UK: Pakt, 2015.

(6) Arduino Foundation, *Introduction*, https://www.arduino.cc/en/Guide/Introduction

(7) Raspberry Pi Foundation, *Raspberry Pi 2 Model B*, https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

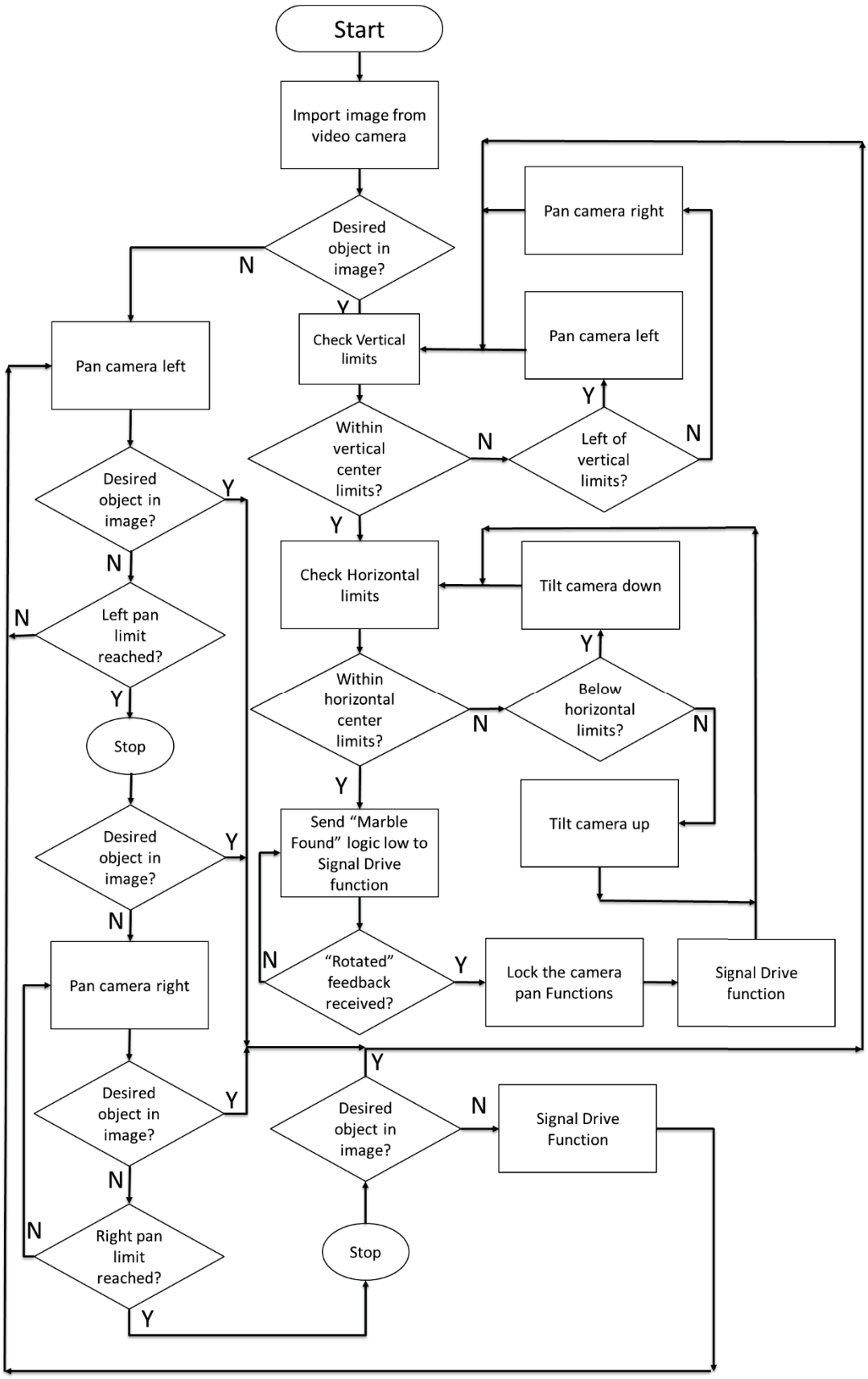



**Figure 10.** Flow chart for the image processing functions.

# Learning to Control First Order Linear Systems with Discrete Time Reinforcement Learning

Eric Nelson
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77843, USA
ejn8411@tamu.edu

Thomas Ioerger
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77843, USA
ioerger@cs.tamu.edu

*Abstract*— **Reinforcement learning (RL) is a powerful method for learning policies in environments with delayed feedback. It is typically used to learn a control policy on systems with an unknown model. Ideally, it would be desirable to apply RL to learning controllers for first-order linear systems (FOLS), which are used to model many processes in Cyber Physical Systems. However, a challenge in using RL techniques in FOLS is dealing with the mismatch between the continuous-time modeling in the linear-systems framework and the discrete-time perspective of classical RL. In this paper, we show that the optimal continuous-time value function can be approximated as a linear combination over a set of quadratic basis functions, the coefficients of which can be learned in a model-free way by methods such as Q-learning. In addition, we show that the performance of the learned controller converges to the performance of the optimal continuous-time controller as the step-size approaches zero.**

## I. INTRODUCTION

The use of reinforcement learning is becoming an increasingly popular choice for learning to control arbitrary systems by minimizing the error of a given objective function. This is because near optimal control policies can be obtained without having to know the underlying system model. If the model of the system is known, the optimal control can sometimes be derived analytically or through the use of dynamic programming and other methods. However, the analytical solutions, even when they can be obtained, are often complicated. This is even more true in the case of higher order dynamical systems. Therefore, this work focuses only on first-order (FO) linear systems. In addition, it is possible for the dynamics of linear systems to vary with time. For example, an airplane's weight changing as the fuel it is carrying is burned. Solutions to these types of systems do exist however they are not the focus of this paper. We only consider the control of first-order linear time-invariant (FO LTI) systems.

The analytical control solutions of FO LTI systems are continuous-time functions that define an optimal trajectory through state-space. However, in many CPSs, control inputs cannot be performed in real-time. For instance, the controls performed by a digital control system are limited by the clock cycle time of the underlying hardware. We assume in this paper that for these discrete time systems a control is selected and is held for a minimum of $\Delta t$ before another control can be selected where the time between successive control inputs is denoted as $\Delta t$.

Reinforcement learning (RL) is a method for learning a policy in a particular environment by exploring the state and action space and receiving rewards after transitioning to a new state. Some RL algorithms, such as value iteration, require knowing a-priori the reward functions and state-transition functions. To remedy this, Q-learning was proposed to be able to learn without explicit knowledge of the reward and state-transition functions [1]. In Q-learning, we attempt to learn a function of both a state and an action instead of attempting to learn a function that gives a value only based on state. The Q-learning algorithm was originally proposed to work on problems with discrete state spaces. Eventually, methods to adapt Q-learning to continuous state spaces were developed in which the state space can be represented by a set of weighted basis functions [2].

In this paper, we show that an optimal controller for a linear system can be learned through the use of reinforcement learning. In addition, we provide insight into how the selection of a $\Delta t$ affects the policies learned by the reinforcement learner. It seems intuitive that a smaller $\Delta t$ will lead to better control policies and reducing the cycle time of the underlying hardware is possible through engineering methods. However, this can significantly increase the cost of the design. The selection of a larger $\Delta t$ will allow for costs to be decreased by allowing slower (cheaper) hardware to be used. Further, we show how the properties of linear systems can be exploited in order to decompose the description of the system into a transient response which affects how the system evolves shortly after a new control input is selected and a long-term response which specifies where the system will come to rest. In order to do so, we have to make a few assumptions. First and as discussed above, we are only concerned with LTI systems. In addition, we assume once a control input has been selected we must hold it for the entire $\Delta t$. We show the reinforcement learner can learn a weighted basis function representation of the value function that approximates it's optimal continuous time counterpart. Finally, we show empirically that control error shrinks with $\Delta t$ and approaches optimal continuous time control in the limit (as $\Delta t \to 0$).

ISBN: 1-60132-433-2, CSREA Press ©

## II. CONTROL OF LINEAR SYSTEMS

### A. State-Space Representation

Modern control theory introduces the idea of the state of a system. In this representation, a set of state variables are chosen and the system is described by how these variables evolve over time. Any FO LTI system with $n$ state variables, $r$ input variables, and $m$ output variable can be written in terms of change in state by the following simultaneous differential equations:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

where $x$ is the state, $y$ is the output and $u$ is the control input. $\mathbf{A}$ is an $n \times n$ matrix that describes the system dynamics. $\mathbf{B}$ is an $n \times r$ matrix that describes how the inputs affect the change in state. $\mathbf{C}$ is an $m \times m$ matrix and $\mathbf{D}$ is an $m \times r$ matrix.

We can solve the differential equation for $x$ by employing the Laplace transform.

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s) \quad (1)$$

Taking the inverse Laplace transform we get:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\, d\tau \quad (2)$$

Therefore, we can see that the internal behavior (state space trajectory) is not, in general, linear and instead the system evolves exponentially as a function of time. These trajectories have a transient phase of rapid response before slowing to a new equilibrium point which we call the long-term or asymptotic response. If we assume that $\mathbf{A}$ is negative definite then as a control is applied, the exponential terms will approach zero as time passes. Eventually, the system will converge to a new equilibrium point as the exponential terms approach zero.

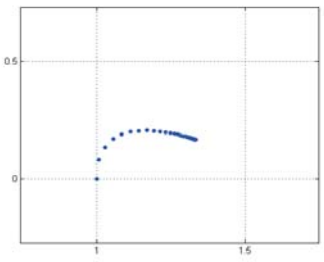



Fig. 1: State space response of the example system to unit step input. The state changes rapidly during the initial period (transient) after a new control input is applied and gradually converges to a new equilibrium point (long-term).

### B. Example

Consider the following first-order linear system:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -1 & 2 & 0 \\ -1 & -4 & 1 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \mathbf{u}(t)$$

The solution can be derived as follows:

$$(s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s+1 & -2 & 0 \\ 1 & s+4 & -1 \\ 0 & 0 & s+1 \end{bmatrix}$$

$$(s\mathbf{I} - \mathbf{A})^{-1} =$$

$$\begin{bmatrix} \frac{2}{s+2} - \frac{1}{s+3} & \frac{2}{s+2} - \frac{2}{s+3} & -\frac{2}{s+2} + \frac{1}{s+3} + \frac{1}{s+1} \\ \frac{1}{s+3} - \frac{1}{s+2} & \frac{2}{s+3} - \frac{1}{s+2} & \frac{1}{s+2} - \frac{1}{s+3} \\ 0 & 0 & \frac{1}{s+1} \end{bmatrix}$$

To solve for $x(t)$ with

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{u}(t) = \begin{bmatrix} U_1(t) \\ U_2(t) \end{bmatrix}$$

where $\mathbf{u}(t)$ is a step function. We now have the parts to evaluate Equation 1. Multiplying the matrices out we get:

$$(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) = \begin{bmatrix} \frac{1}{s+1} \\ 0 \\ \frac{1}{s+1} \end{bmatrix}$$

and

$$(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s) =$$

$$\begin{bmatrix} \frac{1}{s+1} & \frac{2}{s+2} - \frac{2}{s+3} \\ 0 & \frac{2}{s+3} - \frac{1}{s+2} \\ \frac{1}{s+1} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{s} \\ \frac{1}{s} \end{bmatrix} = \begin{bmatrix} -\frac{1}{s+1} - \frac{1}{s+2} + \frac{2}{3(s+3)} + \frac{4}{3s} \\ \frac{1}{2(s+2)} - \frac{2}{3(s+3)} + \frac{1}{6s} \\ \frac{1}{s} - \frac{1}{s+1} \end{bmatrix}$$

Therefore,

$$\mathbf{X}(s) = \begin{bmatrix} \frac{1}{s+1} \\ 0 \\ \frac{1}{s+1} \end{bmatrix} + \begin{bmatrix} -\frac{1}{s+1} - \frac{1}{s+2} + \frac{2}{3(s+3)} + \frac{4}{3s} \\ \frac{1}{2(s+2)} - \frac{2}{3(s+3)} + \frac{1}{6s} \\ \frac{1}{s} - \frac{1}{s+1} \end{bmatrix}$$

so that $\mathbf{x}(t) =$

$$\begin{bmatrix} e^{-t} - e^{-t} - e^{-2t} + \frac{2}{3}e^{-3t} + \frac{4}{3} \\ \frac{1}{2}e^{-2t} - \frac{2}{3}e^{-3t} + \frac{1}{6} \\ e^{-t} + 1 - e^{-t} \end{bmatrix} = \begin{bmatrix} -e^{-2t} + \frac{2}{3}e^{-3t} + \frac{4}{3} \\ \frac{1}{2}e^{-2t} - \frac{2}{3}e^{-3t} + \frac{1}{6} \\ 1 \end{bmatrix} \quad (3)$$

We will analyze the transient and long-term behavior of this system below.

### C. Solving for the Optimal Continuous Time Trajectory

Next, we need to find a policy $\mathbf{u}$ that optimizes some objective. The calculus of variations will prove to be useful in finding such a policy.

Mathematically, we can describe the objective function as:

$$J = \int_a^b f(x, y, y')\, dx$$

where we want to find the function $y$ that minimizes $J$ over the range $[a, b]$ and satisfies initial and final boundary conditions. In this paper, we consider that the goal is to reach a target state $x_d$, and $f(\cdot)$ is the distance squared as a cost function, $f = \mid x_t - x_d \mid^2$ or, for generality, $f = \mathbf{x^T Q x}$ in matrix form, where $\mathbf{Q}$ is a matrix of weights. Using Calculus of Variations, a necessary condition for an extremum function is that it satisfies the Euler-Lagrange equation:

$$\frac{\partial f}{\partial y} - \frac{d}{dx}\left(\frac{\partial f}{\partial y'}\right) = 0$$

The optimal control for a given objective function can be found in a similar manner. This will correspond to how we find an optimal policy in the case of continuous state spaces. For example, take the objective function [3]:

$$J = \int_0^{t_f} \mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)\mathbf{R}\mathbf{u}(t)\ dt$$

First we define the pre-Hamiltonian as:

$$\mathcal{H}(x, u, p, t) = L(x, u, t) + p^T(t) f(x, u, t)$$

$$\dot{p} = \frac{\partial \mathcal{H}}{\partial \mathbf{x}}$$

$$\mathcal{H} = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + \mathbf{p}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u})$$

Pontryagin's minimum principle specifies that the optimal policy is one that minimizes the Hamiltonian at every time $t$ [3]. The Hamiltonian for this example is minimized by setting:

$$\frac{\partial \mathcal{H}}{\partial \mathbf{u}} = 0 = 2\mathbf{R}\mathbf{u} + \mathbf{p}^T\mathbf{B}$$

$$\mathbf{u}^*(t) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{p}(t)$$

where $\mathbf{p}(t)$ is the co-state vector that can be solved for through the use of boundary conditions. This gives us the optimal continuous time policy $\mathbf{u}^*(t)$ that minimizes $J$. Furthermore, the optimal continuous-time trajectory can be derived as $\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}^*(\tau)\ d\tau$.

### D. Solving for the Optimal Continuous Time Value Function

One can solve for the optimal value function by solving the Hamilton-Jacobi-Bellman (HJB) equation [4]. If we define the value function as:

$$V = \int_0^{\infty} \mathbf{x}(t)^T\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T\mathbf{R}\mathbf{u}(t)\ dt$$

where $\mathbf{Q}$ and $\mathbf{R}$ are positive definite matrices. Then we can define a differential equivalent [5]:

$$0 = r(x, u) + \nabla V^T \dot{x}$$

The RHS of this equation is the Hamiltonian of the system.

$$\mathcal{H}(x, u, \nabla V) = r(x, u) + \nabla V^T \dot{x}$$

We can now define the HJB equation as:

$$0 = \min_u \mathcal{H}(x, u, \nabla V)$$

If we assume that the value function is a quadratic function in the Euclidean distance to the goal then we can derive an analytical solution for the value function based on solving the Lyapunov equation $\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q} = 0$ [5].

$$V(x) = \mathbf{x}^T\mathbf{P}\mathbf{x}$$

$$\mathbf{P} = \int_0^{\infty} e^{t\mathbf{A}^T}\mathbf{Q}e^{t\mathbf{A}}\ dt$$

where $\mathbf{P}$ is the solution to the Lyapunov equation. Note that the integral in the definition of $\mathbf{P}$ doesn't need to be solved directly. Instead, $\mathbf{P}$ is the solution of the $n(n+1)/2$ simultaneous equations defined by the Lyapunov equation. Also, note that these results only hold if the origin is the goal. Therefore, if the goal is to control the system to a target equilibrium point, $x_d$, that is not at the origin, then a change of variables where the new origin is defined as $w = x - x_d$ can be carried out using the chain rule:

$$\frac{d\mathbf{w}}{dt} = \dot{\mathbf{w}} = \frac{d\mathbf{w}}{d\mathbf{x}}\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}(\mathbf{K} + \Delta\mathbf{u})$$

$$\dot{\mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{A}\mathbf{x}_d + \mathbf{B}\mathbf{K} + \mathbf{B}\Delta\mathbf{u} = 0$$

$$\dot{\mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{B}\Delta\mathbf{u}$$

where $\mathbf{K} = \mathbf{B}^{-1}\mathbf{A}\mathbf{x}_d$.

## III. REINFORCEMENT LEARNING

Reinforcement learning (RL) is a method of machine learning in which an agent learns a control policy by attempting to maximize the returned rewards for taking an action in a given state [6]. It does this by exploring the state and action spaces and observing the rewards in order to approximate the *value function* which is the expectation of the sum of discounted rewards for starting in state $s$ and following the policy $\pi$. An important assumption made by the RL framework is that state changes are discrete; when an action is taken, a transition to a new state occurs, and the reward is observed, instead of happening continuously in time. If the value function is known, the optimal policy $\pi^*$ can be derived as:

$$\pi^*(s) = \arg\max_{s'} V(s')$$

### A. Markov Decision Processes

Much of the theory of reinforcement learning is based on what are called *Markov Decision Processes* (MDPs) [7]. A MDP is defined on a set of states $S$ and actions $A$ available to an agent. The size of the state and action spaces are traditionally assumed to be finite. The reward function $R(s, a)$ defines the feedback given to the agent for being in state s, taking action a, and ending in state $s'$. Actions can either be deterministic or non-deterministic. If the current state depends only on the previous state and action taken then the problem is said to satisfy the *Markov property*. Given this definition of an MDP, we now define the value function for a given policy (mapping of states to actions) $\pi$ as:

$$V^{\pi}(s) = E_{\pi}[R(s)] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{k+1}]$$

Where $\gamma$ is a *discount factor* that tunes how much the agent should care about future rewards versus the current reward. In other words, the value of being in state $s$ is equal to the expected value of sum of the current reward and all future rewards for being in state s and following $\pi$ until termination.

If the reward and transition functions are known, an MDP can be solved by employing dynamic programming techniques [8]. Value iteration is one such technique in which one can iterate on the value function by taking the max over the actions $V(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')]$. The iterations continue until the LHS of the equation is equal to the RHS. The equation above is called the *Bellman equation*, which is based on the Bellman Principle of Optimality.

### B. Q-learning

Some typical methods of reinforcement learning, such as the value iteration algorithm, require the learner to know some model of the underlying system (i.e. the reward function or state-transition function). However, it is often the case that one does not know these functions. Q-learning can be used to learn policies without having to know a-priori the reward or state-transition functions. Instead of a value function, we now define a Q-function that represents the value of being in state $s$ and taking action $a$:

$$Q^\pi(s,a) = E_\pi[R(s,a)] = E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{k+1}]$$

The Q-function is typically represented as a table called a Q-table where the rows and columns are the finite states and actions. Q-learning can be extended to infinite state spaces by representing Q-functions as the sum of weighted basis functions:

$$\hat{Q}(s,a) = \sum_{i=1}^{N} w_i \phi_i(s,a)$$

This representation requires choosing a set of basis functions that can accurately represent the true value function for a given problem.

## IV. CONNECTING REINFORCEMENT LEARNING AND CONTROL THEORY

Optimal control theory and reinforcement learning share the goal of obtaining an optimal control policy for a particular problem. The difference between the two is the knowledge of the model and assumptions about time. In the optimal control theory case, the control policies are derived exactly with full knowledge of the system and assume continuous time. In the case of reinforcement learning, the model of the system might not be fully or even partially known, and the policies are derived based on interaction with the environment over discrete periods of time, $\Delta t$. In this section, we describe the control policies obtained from both fields in order to derive some insight as to how the two different policies change with respect to $\Delta t$. The questions we address are: 1) Is the value function in the discrete case $V_{\Delta t}$ representable as a linear mapping? 2) How closely would it approximate the continuous time $V^*$ for a finite $\Delta t$?

### A. System Behavior Decomposition

In this section, we prove that a given linear system can be decomposed into a linear function of the transient and long term behaviors of the system.

*Theorem 1:* If $\mathbf{A}$ has both *real* and *distinct* eigenvalues, then each entry in the solution to the state equation (Eq. 2) can be written as the sum of exponentials and constants.

Recall the solution to the state space equations in the Laplace domain given by Equation 1. Note that $x(0)$ and $\mathbf{B}$ are both constant and $\mathbf{U}(s)$ is a step input. Therefore, let us focus on the $(s\mathbf{I} - \mathbf{A})^{-1}$ term first.

Using $|s\mathbf{I} - \mathbf{A}| = 0$, we can obtain the eigenvalues of $\mathbf{A}$, which we assume by the theorem are both real and distinct. We can write:

$$|s\mathbf{I} - \mathbf{A}| = (s - \lambda_1)(s - \lambda_2)...(s - \lambda_n)$$

which is a polynomial of degree $n$.

Let $M_{ij}$ be the matrix obtained by eliminating the $i^{\text{th}}$ row and $j^{\text{th}}$ column from $\mathbf{A}$. $|M_{ij}|$ is called the minor of $a_{ij}$ and has a degree $\leq (n-1)$. Let

$$A_{ij} = (-1)^{i+j} |M_{ij}|$$

be called the cofactor of $a_{ij}$. The adjoint of $\mathbf{A}$ is the matrix in which:

$$adj(A)_{ij} = A_{ji}$$

In other words, the adjoint of $\mathbf{A}$ is the transpose of the matrix of cofactors. Since

$$(s\mathbf{I} - \mathbf{A})^{-1} = \frac{adj(s\mathbf{I} - \mathbf{A})}{|s\mathbf{I} - \mathbf{A}|}$$

We can use Eqs 1, 2, and 3 to write:

$$\frac{adj(s\mathbf{I} - \mathbf{A})_{ij}}{(s - \lambda_1)(s - \lambda_2)...(s - \lambda_n)}$$

$$= \frac{(s\mathbf{I} - \mathbf{A})_{ji}}{(s - \lambda_1)(s - \lambda_2)...(s - \lambda_n)}$$

$$= \frac{(-1)^{j+i} |M_{ji}|}{(s - \lambda_1)(s - \lambda_2)...(s - \lambda_n)}$$

Because the degree of $|M_{ji}|$ is at most $(n-1)$ the degree of the numerator is strictly less than the degree of the denominator.

Therefore, we can use partial fraction decomposition since there are no repeated roots in the denominator and the degree of the numerator is strictly less than the degree of the denominator to get the form:

$$\frac{k_1}{(s - \lambda_1)} + \frac{k_2}{(s - \lambda_2)} + ... + \frac{k_n}{(s - \lambda_n)}$$

Finally, taking the inverse Laplace transform we get:

$$k_1 e^{\lambda_1 t} + k_2 e^{\lambda_2 t} + ... + k_n e^{\lambda_n t}$$

for each entry in $(s\mathbf{I} - \mathbf{A})^{-1}$ as shown below.

$$(s\mathbf{I}-\mathbf{A})^{-1} = \begin{bmatrix} k_{1_{11}}e^{\lambda_{11}t} + ... + k_{n_{1n}}e^{\lambda_{1n}t} \\ k_{1_{21}}e^{\lambda_{21}t} + ... + k_{n_{2n}}e^{\lambda_{1n}t} \\ \vdots \\ k_{1_{n1}}e^{\lambda_{n1}t} + ... + k_{n_{nn}}e^{\lambda_{1n}t} \end{bmatrix}$$

Since $\mathbf{x}(0)$ is a vector of constants for the first half of equation 1 we get:

$$= \begin{bmatrix} c_1k_{1_{11}}e^{\lambda_{11}t} + ... + c_1k_{n_{1n}}e^{\lambda_{1n}t} \\ c_2k_{1_{11}}e^{\lambda_{11}t} + ... + c_2k_{n_{1n}}e^{\lambda_{1n}t} \\ \vdots \\ c_nk_{1_{11}}e^{\lambda_{11}t} + ... + c_nk_{n_{1n}}e^{\lambda_{1n}t} \end{bmatrix}$$

For the second half of equation 1, we have two parts:

$$(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B} = \begin{bmatrix} b_1k_{1_{11}}e^{\lambda_{11}t} + ... + b_1k_{n_{1n}}e^{\lambda_{1n}t} \\ b_2k_{1_{11}}e^{\lambda_{11}t} + ... + b_2k_{n_{1n}}e^{\lambda_{1n}t} \\ \vdots \\ b_nk_{1_{11}}e^{\lambda_{11}t} + ... + b_nk_{n_{1n}}e^{\lambda_{1n}t} \end{bmatrix}$$

For the unit step in each dimension of $\mathbf{U}(s)$:

$$\mathbf{U}(s) = \frac{1}{s}\mathbf{K^T} = \begin{bmatrix} \frac{k_1}{s} & \frac{k_2}{s} & \dots & \frac{k_n}{s} \end{bmatrix}^T$$

we take the inverse Laplace transform to get:

$$\mathbf{U}(t) = \mathbf{U} = \mathbf{K^T}H(t)$$

where $H(t)$ is the Heaviside step function. So we can see that $(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)$ is equal to the matrix for $(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}$. Therefore,

$$\mathbf{x}(t) = \mathcal{L}^{-1}((s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s))$$
$$= \mathcal{L}^{-1}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}(0) + \mathcal{L}^{-1}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)) =$$
$$\begin{bmatrix} C_1k_{1_{11}}e^{\lambda_{11}t} + ... + C_1k_{n_{1n}}e^{\lambda_{1n}t} \\ C_2k_{1_{21}}e^{\lambda_{21}t} + ... + C_2k_{n_{2n}}e^{\lambda_{2n}t} \\ \vdots \\ C_nk_{1_{n1}}e^{\lambda_{n1}t} + ... + C_nk_{n_{nn}}e^{\lambda_{nn}t} \end{bmatrix}$$

where $C_i = k_i(b_i + c_i)$ and $i$ is equal to the row index. Therefore, as long as the eigenvalues of the matrix $\mathbf{A}$ are both *real* and *distinct* the solution to the state equation can be written as a sum of exponential functions and constants.

If we group the exponential terms together, they form what we call the transient response matrix $\mathcal{T}$ of the system. The constants form the long term response matrix $\mathcal{N}$ of the system. The $\lambda$ terms in the transient response matrix are the time constants of the system. We can now define:

$$x(t) = \mathcal{T}(t) + \mathcal{N}$$

For the example problem in section II, separating exponentials from constants in equation 3, we have:

$$\mathcal{T} = \begin{bmatrix} -e^{-2t} + \frac{2}{3}e^{-3t} \\ \frac{1}{2}e^{-2t} - \frac{2}{3}e^{-3t} \\ 0 \end{bmatrix}, \quad \mathcal{N} = \begin{bmatrix} \frac{4}{3} \\ \frac{1}{6} \\ 1 \end{bmatrix}$$

The smallest time constant is given by $\tau = \frac{1}{3}$. Thus, the system reaches about 37% of the way to the new equilibrium point $\begin{bmatrix} \frac{4}{3} & \frac{1}{6} & 1 \end{bmatrix}^T$ in about 1/3 sec.

### B. Response of LTI Systems to Actions

Unlike general applications of RL, we can derive a transition function from knowledge of a linear system. For any action held constant for $\Delta t$, we can predict the discrete response $x_t \rightarrow x_{t+1}$. The previous section allows us to see that given any step input we can decompose the system into a set of constants and exponential functions. In this section, we will show that we can derive the *state transition function* of an LTI system given the matrix A and a $\Delta t$ over which a constant action $u$ is held. In addition, we show that it is a linear transformation in the state space of the system. We start by defining $\Delta x$ as the difference in state at time $t$ from the initial state.

$$\Delta\mathbf{x} = \mathbf{x}(t) - \mathbf{x}(0)$$

We then plug in the solution for $\mathbf{x}(t)$.

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \mathcal{L}^{-1}((s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s))$$
$$\Delta\mathbf{x} = e^{\mathbf{A}t}\mathbf{x}(0) + \mathcal{L}^{-1}((s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)) - \mathbf{x}(0)$$
$$\Delta\mathbf{x} = (e^{\mathbf{A}t} - \mathbf{I})\mathbf{x}(0) + \mathcal{L}^{-1}((s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s))$$

Since $\mathbf{U}$ is a step input we can represent it as a constant $\mathbf{U}$.

$$\Delta x = \mathbf{M}(\Delta t)\mathbf{x}(0) + \mathbf{L}$$

where

$$\mathbf{M}(\Delta t) = (e^{\mathbf{A}\Delta t} - \mathbf{I})$$

and

$$\mathbf{L} = \mathcal{L}^{-1}((s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U})$$

Therefore, for a given $\Delta t$ the matrix $\mathbf{M}$ and the vector $\mathbf{L}$ are constants, meaning we can predict, for a given state, $\Delta t$ and control input, our next state discrete $x_{t+1}$. We can also see how this transformation is a simple linear transformation of coordinates in state space (given a fixed $\Delta t$) as shown in Figure 2.

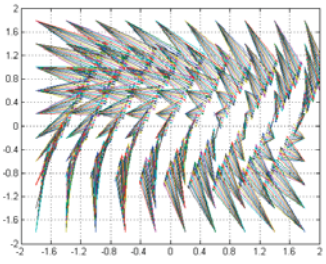


Fig. 2: A depiction of the linear mapping for an input action for the example problem. Each vector shows the result of taking the same action from different coordinates in state space and holding it for $\Delta t = 0.2$.

So in theory, since we can extract the transition function from knowledge of the dynamics, then it should be possible

learn a control policy (for a given reward function) using a dynamic programming technique such as value iteration. However, our goal is to show that a controller can be learned for a linear system without requiring the model (i.e. using Reinforcement Learning).

### C. Approximating the continuous-time value function

Although reinforcement learning is often presented in terms of discrete state spaces, it can be extended to continuous state spaces by representing the value function as a linear combination over a set of continuous basis functions [2]. That is, given a basis set $\phi_1(s)$ to $\phi_n(s)$.

$$V = \sum_{i=1}^{n} w\phi_i(s)$$

where $w_i$ is a weight for each $\phi_i$. Two important questions in applying discrete-time RL to learning control of a continuous-time LS are: 1) can the continuous-time value function be approximated as a linear combination?, and 2) over what basis set? Since the Lyapunov equation (above) shows that if the reward is quadratic ($r = \mathbf{x^T Q x}$, then the optimal continuous-time value function can also be expressed as a quadratic function $V^* = \mathbf{x^T P x}$. Expanding this matrix expression out as a polynomial, $V^* = \Sigma_i \Sigma_i P_{ij} x_i x_j$. Therefore, a natural set of basis functions is the $n(n+1)/2$ combinations of products of the state space variables (up to second-order). For the example in the previous section, we would choose: $[\phi_1 ... \phi_6] = [x_1^2,\ x_1x_2,\ x_1x_3,\ x_2^2,\ x_2x_3,\ x_3^2]$

*Theorem 2 (Representation Theorem):* The optimal continuous-time value function $V^*$ is representable as linear combination over a quadratic basis set, $V^* = \sum_{i=1}^{n(n+1)/2} w_i\phi_i(s)$ where $[\phi_1 ... \phi_{n(n+1)/2}] = [x_1^2, x_1x_2 ..., x_ix_j, ...x_n^2]$.

We show below that the optimal discrete-time value function $V_{\Delta t}$ can be made to approximate $V^*$. Thus the Representation Theorem shows that $V^*$ can be learned as a target by using discrete-time methods (e.g. Reinforcement Learning) by learning weights over a quadratic basis set. However, as shown in the next section, the accuracy of approximation will depend on the size of the time step, $\Delta t$.

### D. Effect of $\Delta t$

If we assume continuous control inputs that are bounded by some constant $\mu$ (i.e. $|U| \leq \mu$) then a *reachable* region from the current state is defined given a $\Delta t$. This is illustrated in Figure 3 as the light gray region. A state $x_2$ is said to be *reachable* from $x_1$ if there exists a control input $u(t)$ such that $|u(t)| < \mu$ that can transfer the system from $x_1$ to $x_2$ in finite amount of time. It should be noted that the shape of the region is dependent on the dynamics of the system and is not necessarily circular. The optimal trajectory, $x^*(t)$, is represented as the solid black line in the figure and a constant input trajectory for a given action is shown as a dotted line. The dotted line represents choosing an action and holding it for $\Delta t$. It is curved because of the non-linear, transient response of the system.

*Theorem 3:* Assuming bounded, continuous control inputs, there exists an action that intersects with either the optimal trajectory along the boundary of the reachable region or the goal if it is inside the reachable region and this action belongs to the optimal policy.

This can be proven by contradiction. The policy we learn, once converged, has the property that the action chosen for a given state maximizes the value function. That is

$$\pi^*(s) = \arg\max_{s'} V(s')$$

which means that $V(s') \geq V(s)\ \forall s \in S$ where $S$ is the full state space. In addition, the optimal trajectory has the property that it maximizes the optimal value function for every point along optimal trajectory. That is: $V^*(s) \geq V(s')\ s' \neq s$. Therefore, if we choose an action that leads to a state not at the intersection of the optimal trajectory and the reachable region's boundary then its value is less than the optimal value. This violates our assumption that $V(s') \geq V(s)\ \forall s \in S$ because we could choose an action with a higher value by choosing the action at leads to the intersection point.

Next, we show that as $\Delta t \to 0$, we get better control policies. For a smaller $\Delta t$, the reachable region is smaller which implies the maximum distance away from the optimal trajectory is also smaller. The error of the policy for a given $\Delta t$ is defined by the difference of the integral of the distance to the goal for the trajectories produced by both policies.

An important goal is to show that the value function learned by reinforcement learning will converge in the limit (as $\Delta t \to 0$) to the optimal value function for continuous time control,

$$V^* = \int_0^\infty (\mathbf{x(t)} - \mathbf{x_d})^2\ dt$$

Although the discrete-time value function is conventionally defined $V = \Sigma \gamma^i r_t$, in order to get it to converge to $V^*$, we modify by setting $\gamma = 1$ and multiply by $\Delta t$:

$$V_{\Delta t} = \Delta t\ \Sigma_{t=0}^{\infty}\ r_t = \Delta t\ \Sigma_{t=0}^{\infty}\ (\mathbf{x(t)} - \mathbf{x_d})^2$$

Since for any time step $\Delta t$, there is always a constant action that can move the system to the same point on the continuous time trajectory, as argued above, then the rewards $r_t = (\mathbf{x(t)} - \mathbf{x_d})^2$ will match at these discrete points. Thus, the discrete-time value function is simply the numerical approximation of the continuous-time value function, $V^*$.

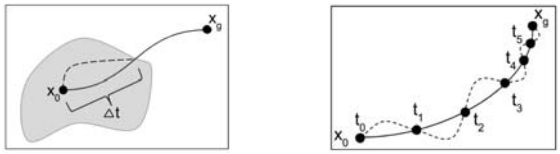
$$V^* = \int_0^\infty (\mathbf{x(t)} - \mathbf{x_d})^2\ dt \approx \Sigma_{t=0}^{\infty}\ \Delta t (\mathbf{x(t)} - \mathbf{x_d})^2$$

$$\lim_{\Delta t \to 0} V_{\Delta t} \to V^*$$

Furthermore, we can we put a bound on the error in the value function. The error is a function of the time step size (as with any numerical quadrature), as well as the maximum deviation between the optimal continuous-time trajectory and the constant-input trajectory between the discrete time points, where the coordinates coincide. This is bounded because the

amount the constant-input trajectory can diverge from the continuous time path within $\Delta t$ is limited by the dynamics of the system, which will force them to follow similar paths.

The response to a given input, and therefore the error, is dependent on the underlying system dynamics and it is also dependent on how quickly a system responds to a new input. The smallest system time constant, $\tau$, defines how rapidly the system responds to a new control input. Therefore, we propose that a requirement for the selection of a $\Delta t$ is that it be less than $\tau$ in order to ensure that these rapid responses can be controlled. At $\Delta t = dt$ the optimal trajectory will equal the approximate trajectory through state space.




(a) (b)

Fig. 3: Differences between the optimal continuous-time trajectory and the trajectory produced by a sequence of discrete inputs. Solid line shows continuous-time trajectory. Shaded region shows the reachable space within $\Delta t$. Dashed line shows trajectory by a sequence of constant inputs.

## V. EXPERIMENTS

We analyzed and tested a stable, 2-dimensional system with the dynamics defined by the matrices $\mathbf{A} = [[-1\ \ 2], [-1\ \ -4]]$ and $\mathbf{B} = [1\ \ 1]^T$, and evaluated the effect of $\Delta t$ on policies learned by Q-learning. This system is stable because $\mathbf{A}$ is negative definite. The time constants of the system are: $\tau_1 = \frac{1}{3}$ and $\tau_2 = \frac{1}{2}$. The system was implemented in MATLAB using fourth order Runge-Kutta methods to simulate the system dynamics with a time step of $\Delta t = 0.001$ seconds. The goal is to show that the accuracy of the learned controller increases as $\Delta t \to 0$.

We used the Q-learning algorithm to learn the weights of the quadratic functions of the state variables. The system is set in a random initial location for an episode and uses an $\epsilon$-greedy action selection mechanism [9] to explore the state space. The actions available to the learner were those between -10 and 10 with steps of 0.1 and the goal was to reach (0,0).

As shown in Figure 4, as $\Delta t \to 0$ the value estimate increases which represents a better control policy. This figure was generated by, first, learning for each $\Delta t$ until the Q-function converged which we defined as 10000 weight updates with maximum weight changes of less than $10^{-12}$ or 5000 episodes completed. Second, we tested each policy by setting $x_0 = (2,2)$ and running the system until the goal is reached. The squared distance to the goal accumulated along the trajectory is the value presented in the graph. Note that $V_{\Delta t}$ plateaus below the lowest time constant of the system. This suggests that there are diminishing returns to reducing $\Delta t$ further than the start of the plateau and that a $\Delta t$ of 0.01 would be adequate to learn a near-optimal controller given the dynamics of the system.

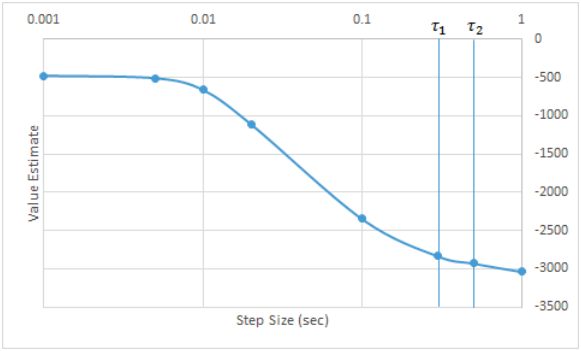



Fig. 4: The value estimates from the learned policies as a function of $\Delta t$.

## VI. DISCUSSION

In this paper, we have discussed how it is possible to decompose a system into a transient response and a long term response. The decomposition provides some insight as to the selection of a control time for a given learner and enables one to use methods such as value iteration to find an optimal value function $V_{\Delta t}$. We have shown that for a quadratic cost (or reward) function the optimal value function is also quadratic and it is representable by a weighted linear combination of a quadratic basis set. Finally, we have shown that as $\Delta t \to 0$ the approximation to the optimal value function is more accurate and therefore we get better control policies. This idea was corroborated by the tests on an example dyanmical system where we see that the $\Delta t$ should be less than the smallest time constant of the system in order to generate a near-optimal policy.

## REFERENCES

[1] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[2] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, Dec. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=945365.964290

[3] W. L. Brogan, *Modern Control Theory (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[4] M. Johnson, R. Kamalapurkar, S. Bhasin, and W. E. Dixon, "Approximate-player nonzero-sum game solution for an uncertain continuous nonlinear system," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, no. 8, pp. 1645–1658, 2015.

[5] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Reinforcement Learning and Optimal Adaptive Control*. John Wiley & Sons, Inc., 2012, pp. 461–517. [Online]. Available: http://dx.doi.org/10.1002/9781118122631.ch11

[6] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996. [Online]. Available: http://people.csail.mit.edu/lpk/papers/rl-survey.ps

[7] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[8] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[9] M. Tokic and G. Palm, *KI 2011: Advances in Artificial Intelligence: 34th Annual German Conference on AI, Berlin, Germany, October 4-7,2011. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax, pp. 335–346. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24455-1_33

# Using Extensible Components to Address Semantic Mapping Deficiencies in Cyber-Physical Systems

Michael Jonas, Ashraf Gaffar
[1]CIDSE, Arizona State University, Tempe, AZ, USA
[2]CIDSE, Arizona State University, Mesa, AZ, USA

**Abstract** — *Cyber-Physical Systems (CPSs) often integrate existing components to solve well-known problems such as networking, coordination, or planning. Unlike pure software systems, when these components are integrated there are often physical concerns which cannot be ignored. Addressing these concerns requires mapping the CPS semantics to the component semantics. Semantic misalignments can render a system prohibitively complex, costly, or dangerous. This work demonstrates this problem using planning semantics and provides an extensible alternative which enables bringing the CPS model to the component intact.*

## I. Introduction

Cyber-Physical Systems (CPSs) by definition contain a cyber and physical component. These systems not only exist in the physical world but affect the world as part of their core functionality by means of actuation. The world in turn affects the computation of the cyber domain via sensing. Between this sensing and actuation the cyber component of the CPS makes sense of the world and via some algorithmic process decides on a course of action which fulfills the system's purpose as shown in Fig 1. This general approach has been used by cognitive architectures since their foundation [1-3].

The purpose and nature of the problem each cyber-physical system attempts to solve will be different, resulting in different solutions. However, many of the solution subtasks are already established in different fields. Many CPSs include foundational components such as statistical libraries, planning libraries, and model checking tools [4-7]. These components are used because they are well-established and CPS owners hope to lower development costs.

For these components to be useful, a predictive model of the physical world must be provided to the system. The system relies on an accurate understanding of the semantics of the components used. An accurate predictive model can be extremely complex. A single pure simulation can take entire data centers long hours to compute to check a single solution; a luxury most cyber-physical systems don't have even before considering prohibitively large solution spaces. A working CPS relies on abstractions to make the modeling cost manageable. These abstractions are specific to each domain and are best known by the experts of that domain. These experts are the owners of the CPS but are merely users of the framework or component. Domain experts bring with them a ubiquitous language specific to the domain with a distinct set of concepts, structures, relationships, intuitions about how the domain behaves, and an efficient means of modeling that domain. In order to use the borrowed component the domain expert has to map the domain semantics to the component semantics as shown in Fig 2.

While domain semantics are typically rich and mature in the domain language, the component semantics are often quite restrictive, relying on limited semantics.

For example, planners are specialized tools that are useful in finding plans to achieve goals. CPSs with domains where the systems response to input is a sequence of behavior rather than reflexive can find planners useful for deriving this sequence and managing a problem which is inherently NP-complete. The semantics planners use are based on state transition systems [5, 8, 9]. Each state-transition system is a 3-tuple $\Sigma = (S, A, \gamma)$, where $S = \{s_0, s_1, \ldots\}$ is a set of states, $A = \{a_0, a_1, \ldots\}$ is a set of actions which cause transitions between states and $\gamma : S \times A \rightarrow 2^S$ is a state-transition function. Plainly, the number of possible states even for simple systems necessitates specialized algorithms. In classical planning, these algorithms rely on numerous assumptions which are problematic and untrue for many physical models used by CPSs.

- Assumption A1 (Finite Σ). The system Σ has a finite set of states.
- Assumption A2 (Static Σ). The system Σ is static and has no internal dynamics; it stays in the same state until an action is applied.
- Assumption A3 (Attainment Goals). Goals are defined as reaching one of a collection of states. Planning is the process of finding a sequence of state transitions that ends at one of the goal states. This definition excludes states to be avoided, constraints on state trajectories, and utility functions.
- Assumption A4 (Sequential Plans). A solution to a planning problem is a linearly-ordered finite sequence of actions.
- Assumption A5 (Implicit Time). Actions are instantaneous state transitions with no explicit definition of time in the action or state.

ISBN: 1-60132-433-2, CSREA Press ©

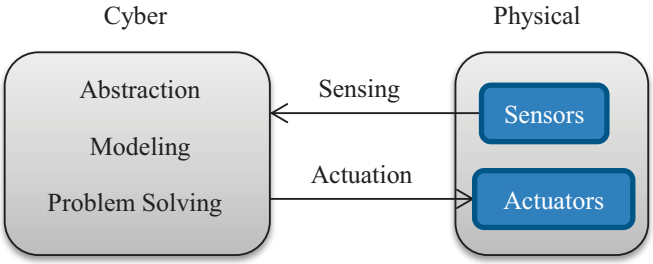



Fig. 1. A high-level overview of a cyber-physical system and its interactions with the physical world.

While the planning community has improved on many of these assumptions since the original planning competition in 1998 these assumptions still exist in spirit in many ways today.

Components are valuable because of the algorithms they provide. However, all algorithms must address an agreed upon structure to act on them. For example in strongly typed languages the type must be defined before an algorithm can do very much with a reference. These type definitions are a necessary constraint of both the component that defines them and the user of the component which must map domain semantics to the component semantics before an algorithm can function.

Semantic misalignments are possible when this mapping occurs. An algorithm which uses a structure while having an incorrect or incomplete understanding of it can result in incorrect behavior leading to system failure. On the other hand if the purpose of the algorithm requires information which the structure is insufficient to express the algorithm cannot fulfill its purpose. For example, consider a tic-tac-toe solver being provided a 3-d variant of the game. The types used by a 2-d solver are probably incompatible with the new problem and the component is useless until its structures are updated.

These structures are not merely convention. They organize information for the algorithm and define with precision the forms that information can take. The meaning of this information is encoded by an author into an algorithm by logically consistent usage with respect to a purpose.

When domain experts wish to use a component they must map the domain semantics to the component semantics. In doing so, the assumptions of the component can become problematic in two different ways.

- If these assumptions cause a resolution loss, whatever problem is solved by the component is rooted in a cyber model inconsistent with the physical world. This can render a system unsafe.
- If the assumptions generalize some relationship, this can cause an explosion in the problems difficulty.

We demonstrate both possibilities in the example domain which follows.

This work provides two significant contributions. First, we introduce a new planning domain derived from a real cyber-physical domain which, while being mathematically concise, is problematic for state-of-the-art planning semantics. We shed light on the reasons for this difficulty and argue that the constraints of planning semantics are the root cause rather than the inherent difficulty in the problem. Second, we provide semantics for a state-search structure and model independent algorithm which enables CPS domain authors to bring their models to the planning process intact.

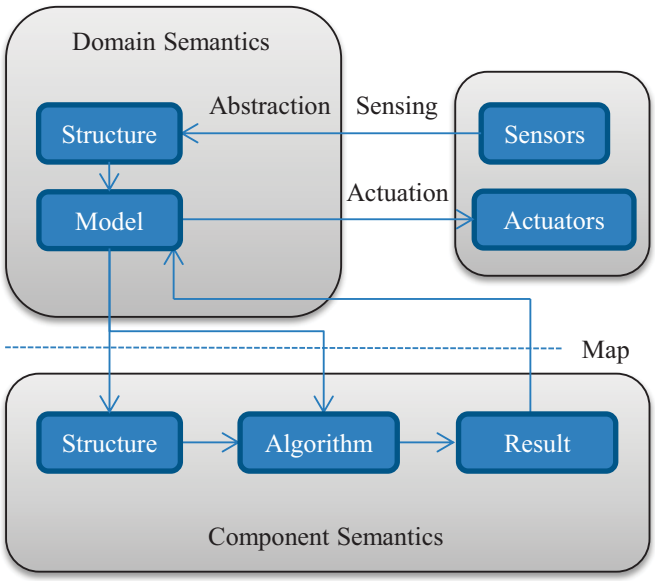



Fig. 2. Domain experts must map or translate domain semantics to a component's semantics and back to use a component.

## II. Example Domain

This section will demonstrate the problems caused by semantic misalignment using an example domain. We use a simplified data center domain, first without cyber-physical interactions, then with them to demonstrate the problems that arise when mapping domain semantics to planning semantics.

Without cyber-physical considerations the domain is a simple task scheduling problem. There is a set of jobs which a data center must run within their deadline Service Level Agreement (SLA). Task scheduling is a well-established computer science problem which requires little elaboration. The model for this requires the following:

- A list of jobs which need completing, the time they take to complete, and the resources they require.
- A constraint which states that each job must be completed before a deadline time.
- A list of chassis with resources.
- An action which schedules a job to a chassis and consumes resources until the job is completed.

Many scheduling algorithms have been developed for similar problems but they mostly only consider the cyber aspects of the problem. The focus of this paper is on a combined cyber-physical solution. We incorporate a physical thermal model which introduces new complex dependencies. Accounting for this physical model significantly changes the characteristics of a good solution. Thermally ignorant task scheduling algorithms tend to pack the job schedule as early as possible in time. This is optimal when the physical consequences of doing so are unconsidered and schedule slack is the only concern. However, when the physical aspect of

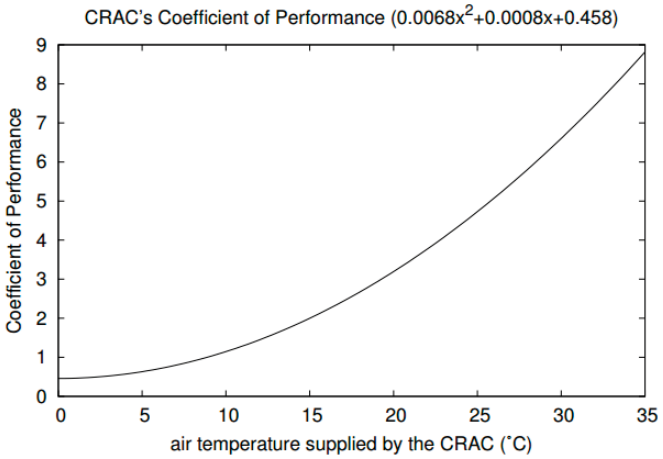



Fig. 3. Coefficient of performance (CoP) of a typical AC unit [11].

energy is considered such a system is demonstrably wasteful. This has led to a body of work on energy-aware scheduling.

For this example we will use a simplified version of our previous work on thermal modeling [10]. The only physical aspects to consider are the air conditioner (AC) efficiency vs. temperature, the maximum temperature of the equipment, and the time delay between air leaving the AC and arriving at a chassis.

AC efficiency is defined in terms of coefficient of performance (COP).The COP of the AC is the amount of heat removed per energy consumed by the AC.

$$\text{COP} = \frac{\text{Heat Removed}}{\text{Work Consumed By AC}}.$$

With a COP of 1, fifty percent of the energy expended by the data center will be the cost of cooling. Simply note that the lower the return temperature and the bigger the delta, the more energy is spent per Joule to remove it. In our data center example if the set of jobs produce an invariant total Joules of energy the cost of removing that energy from the system is higher if the air is being cooled more.

When scheduling algorithms like Earliest Deadline First (EDF) frontload all of the jobs this super-heats the air during that time. The air must then be cooled at a lower COP in order to keep equipment below its maximum temperature. What was previously optimal from a performance perspective is instead pessimal with respect to energy efficiency. This is a problem that can only be addressed if the physical aspects of the problem are considered during scheduling. Hence, energy-aware scheduling requires a physical model in addition to the cyber model already presented. An energy-aware system allows data center owners to improve energy efficiency and save money.

For this simplified model we assume all air arriving at a chassis comes from the air conditioner (AC). We model the temperature of a chassis at a given time, $T_i(t)$, as the average temperature of the air from the AC, $T_{AC}(t)$, over some span of time $t$-$t_{is}$ to $t$-$t_{ie}$ where $t_{is}$ and $t_{ie}$ are the start and end time offsets for the integral bound plus an amount of heat added by load from jobs, $L_i(t)$.

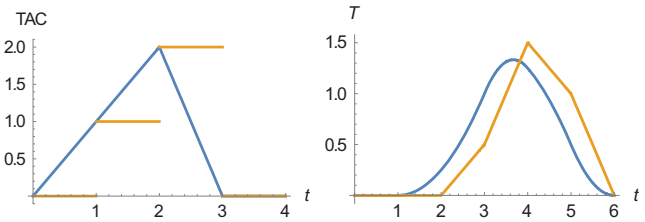



Fig. 4. a. A cooling function $T_{AC}$ and its discretization with a $\Delta$=1 discretization period. b. The resulting $T_i$ curve using Equation 1 assuming no load.

$$T_i(t) = L_i(t) + \frac{\int_{\tau - t_{is}}^{\tau - t_{ie}} T_{AC}(\tau)\partial\tau}{(t_{ie} - t_{is})} \quad (1)$$

Each chassis has a maximum temperature it must stay below. The AC temperature can be adjusted to an arbitrary function, $T_{AC}(t)$, to lower the chassis temperature to keep it below this threshold but lower temperatures increase costs super-linearly. The planner therefore seeks to schedule jobs such that all job deadlines are met and AC cost is minimized.

In devising a solution for this problem we first attempt to map this physical model to planning semantics. This requires the following additions to the previous cyber model.

- $T_{AC}(t)$ must be set by a new action.
- $T_i(t)$ must be calculated using Equation 1.
- A heat load value, $L_i(t)$, must be associated with each job.
- A maximum temperature, $T_{MAX}(i)$, is associated with each chassis.
- A constraint is added which states that each chassis temperature must be below its maximum temperature at all times.
- The job scheduling action updates $L_i(t)$ when the job begins and ends.
- A goal metric is added to minimize total energy.

On trying to implement this model in any planning language the first problem encountered is that the calculus required for the temperature calculation of Equation 1 is not supported in a continuous manner.

Some languages like PDDL+ allow a description of continuous change of the form $\partial V/\partial t = f$. An example in PDDL+ is shown below [12].

```
(:process charging
  :parameters (?r - rover)
  :precondition (and (<= (charge ?r) (capacity ?r))
                     (in-sun ?r)
                     (charging ?r))
  :effect (increase (charge ?r)
                    (* #t (charge-rate ?r))))
```

This process is modeling battery charging in the rover domain. This is a standard domain used in the international planning competition (IPC) based on the challenges of the Mars rover. The Mars rover CPS uses a planner component as

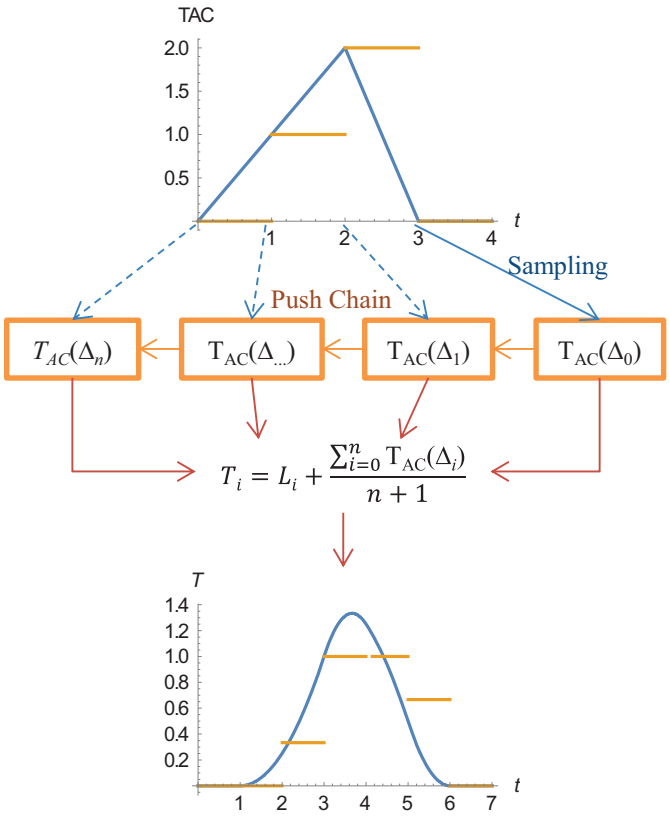



Fig. 5. Synchronization and discretization mechanism enabling Non-Markovian description of Equation 2.

part of its solution which needs to account for the charge of its batteries. The process above states that while the rover, `?r`, has remaining capacity that could be charged, is in the sun, and is charging, the amount of charge is continuously increased by the charge rate.

This looks promising but the formula for *f* cannot refer to values at any time other than the present and the operators it can include are limited to the arithmetic set $(+, -, \times, \div)$. Two approximations are necessary because of this.

- The current state must contain all values necessary for the calculation. This discretizes $T_{AC}$ by sampling its value at distinct time points rather than modeling it continuously.
- The calculation of $T_i$ must be approximated using a summation shown in Equation 2.

$$T_i = L_i + \frac{\sum_{i=0}^{n} T_{AC}(\Delta_i)}{n+1} \quad (2)$$

These approximations require some complex mechanics shown in Figure 5. An ordered set of sample variables, $T_{AC}(\Delta_i)$, are created to hold historical AC temperature values that are sampled at a regular discretization period, $\Delta$, by an artificial modeling action. To maintain semantic consistency, past temperature values are pushed from one variable to the next, $T_{AC}(\Delta_{m+1}) = T_{AC}(\Delta_m)\ \forall 0 \le m \le n$, resulting in $n+1$ variable modifications per $\Delta$ as time progresses. These sample variables can then be used to calculate $T_i$ using Equation 2. These approximations come with several unwanted consequences.

First, since no optimal solutions exist for which $T_{AC}$ is discretized, no optimal solutions can be found with these planning semantics. The optimal solution cannot even be expressed as a plan using state of the art semantics congruent with assumption A4 for basically the same reason a polynomial function can only be approximated by line segments.

Further, as shown in Figure 5, discretizing $T_{AC}$ leads to modeling precision loss which carries over into $T_i$. The $T_i$ calculation from Equation 2 compounds this problem. Underestimating $T_i$ can cause overheating which can cause equipment damage and SLA violations from equipment shutdowns if the inaccurate model is acted upon. Even if an acceptable level of precision loss can be found, this is an undesirable outcome of nothing more than semantic mapping. It is impossible to represent this domain accurately or produce an optimal solution using state-of-the-art planning semantics.

Second, the net result of the process shown in Figure 5 is an explosion in the number of variables which model $T_{AC}$ from 1 to $n+1$. Since the correlation between these variables is lost in translation, they seem independent to proof-based heuristics and such approaches become quickly overrun. Planners which consider combinations of AC temperatures independently are effectively faced with a more complex version of the NP-Complete knapsack problem.

This observation reveals two opposing forces. First, if the discretization period is too large unacceptable precision loss renders the system dangerous or unusable. If the discretization period is too small the variable count explosion renders the system unusable.

For a sense of scale consider several real values from our measured data center. If the heat exceeds the maximum temperature for more than about a second the hardware safety will kick in and the equipment will shut down. When this happens, data integrity can be compromised. At best, restarting the machine and recovering the work takes several minutes during which the performance of the system has suffered. This can cause punitively expensive SLA violations if there was insufficient slack in the schedule to recover from the sudden loss of a machine. This implies the system requires a $\Delta$ of less than 1 second.

The number of discretization windows,$n$, is on the order of minutes. In the measured data center air could take about 10 minutes to cycle which yields n≈600. Solving a problem of this scale is already difficult. The full problem is even worse though. In the full domain model the calculation for $T_i$ considers air recirculation between chassis as well as the AC which would increase this value to $j(n+1)$ where j is the number of thermal points of interest. Large data centers can have tens of thousands of chassis, yielding millions of independent variables. Polynomial time solutions are quickly overrun.

# III. Alternative Approach

After this treatment this domain may seem quite difficult to model, but most of the difficulty arose as artifacts of semantic mapping and algorithms limited to a rigid set of semantics. There are simple, intuitive solutions we could consider if the planning semantics were more extensible and the planning process facilitated dependency injection for control to benefit from our intuition.

Intuitively, a good solution to this domain is one that keeps the load among chassis roughly equal so that when one chassis is cooled all chassis benefit without overcooling. Furthermore, due to the super-linear cost of cooling it is more efficient to spread jobs out over time, cooling just as much as is needed to meet the thermal constraint of the chassis.

Declarative planning semantics could be formulated to address the semantic mapping problems demonstrated in section II. However, that would only serve as an incremental nudge to the limitations experienced by cyber-physical system owners and any polynomial time solution will become non-performant as the domain scales.

We observe that domain semantics are complex in a myriad of ways that exceed component semantics. If semantic misalignment is the cause of problems then models need to be used by components intact. This requires extensible components. We provide a model-independent planning algorithm below which exemplifies this point.

This requires important divergences from the typical approach:

- Where possible, the component should be model-independent.
- Where not possible, allow knowledge injection to augment a limited set of semantics.
- Where knowledge injection within the function is not possible, allow bypassing the function if the component consists of multiple functions.

The modeling semantics of the JPDL framework serve as our baseline [13]. An API implementing these semantics can create classes for States, State Variables, Effects, Constraints, and Goals with reserved semantics for a state-space planning algorithm defined below. Domain authors create subclasses to define their model. The methods reserved by these classes can implement arbitrarily complex behavior while interfaces can be used enable an extensible algorithm.

In this case we link the modeling methods to a Mathematica library. The contribution function of a scheduled job, *s*, is just an indicator function of the job's heat over a range. $L_i(t)$ is a sum of these contribution functions. $T_{AC}(t)$ is an arbitrary function which the planner can set as an Effect. $T_i$ implements Equation 1. The definitions below were used to generate the continuous graphs in Figure 4 and 5.

```
LBase[s_] = Piecewise[
{{0, 0 ≤ t < s[StartTime]},
{s[Job][Heat], s[StartTime] ≤ t < s[StartTime] + s[Job][Runtime]},
{0, s[StartTime] + s[Job][Runtime] ≤ t }}]

L[t_] = Plus[Map[LBase, S]]

TAC[t_] = Piecewise[{{0, t < 0}, {t, 0 ≤ t < 2},
                     {-2t + 6, 2 ≤ t < 3}, {0, 3 ≤ t}}]

T[x_] = L[x] + Integrate[TAC[t], {t, x-ts, x-te}]/(ts-te)
```

The system has two constraints. First, all jobs exist in the schedule sufficiently before their deadline. Second $T_i(t) < T_{iMax}(t)\ \forall i,t$. The goal of the system is to schedule all jobs without violating constraints. This concludes the modeling portion.

Next we define the semantics for a state-space planning algorithm we call the Planning Branch algorithm which uses the above definitions.

A **World Model** $W$ is a tuple $\{P_v, S, E_L, C_L, G_L\}$ consisting of a paired Identity and Primitive set $P_v$, a State S, an ordered pending Effect set $E_L$, a Constraint set $C_L$, and a Goal set $G_L$. It contains all information required for backtracking and exploring planning branches.

A **Decision Epoch Pattern** $P$ is a tuple $\{P_v, V, f(P_v, V, W)\}$ that uses a paired Identity and Primitive set $P_v$, a set of State Variables $V$, and Boolean function $f(W)$ that takes a World Model $W$ and to evaluates whether a pattern contained within $f$ with parameters $P_v$ and $V$ is matched by $W$. When this pattern is matched a decision epoch occurs as described in the Planning Branch algorithm below.

A **Checkpoint** $C$ is a tuple $\{W, P\}$ consisting of a World Model $W$ and a Decision Epoch Pattern $P$ which triggered it. An initial decision epoch occurs after the problem is instantiated during which $P$ is a reserved type. Otherwise, Checkpoints are created when $P$ is matched by $W$.

A **Checkpoint Link** $L$ is a tuple $\{E_L, P_L\}$ consisting of a set of Effects $E_L$ containing planned actions and a set of Decision Epoch Patterns $P_L$ which are used to trigger the next decision epoch. Checkpoint Links are created by Planning Heuristics during decision epochs.

A **Link Tree** $T$ is a tuple $\{C, T_C, L_P, T_P\}$ of a Checkpoint value $C$, a set of children Link Tree nodes $T_C$ each associated with a Checkpoint Link $\{L, T_C\}$, a parent Checkpoint Link node, $L_P$, and a parent Link Tree node, $T_P$, which provides an easily traversable structure for planning heuristics.

The Link Tree is a public structure which tracks search history both for the planning heuristic and for any surrounding code to analyze.

A **Planning Branch** $B$ is a tuple $\{T, L\}$ consisting of the Link Tree node $T$ to branch from and the Checkpoint Link $L$ to branch with.

A **Planning Heuristic** $H$ is a set of functions $f_C(C, W, P)$ and $f_B(B, W, C)$ that return a Planning Branch to explore next. The Checkpoint function, $f_C$, responds to a Decision Epoch Pattern being matched by World Model. The Backtrack function, $f_B$, responds to Constraint violations.

The Planning Branch algorithm is detailed in Figure 6. In our API, the domain author creates subclass implementations for most of these symbols. The base classes are semantically important for the implementation of the Planning Branch algorithm but whatever functionality they provide can be extended using object-oriented paradigms to support models of arbitrary complexity. The algorithm is model-independent.

1. $B$:=$H.f_C(C, W, InitialP)$.
2. Initialize $W$ to explore $B$.
   **if** $B$ = null **then** fail **end if**
   $T$ := $B.T$
   $W$ := clone$(T.C.W)$
   $W.E_L$ := $B.L.E_L \cup W.E_L$
3. Explore B.
   **for all** $e \in E_N$ **do** $W$:=$e.f(W)$ **endfor**
   **for all** $c \in W.C_L$ **do**
   **if** $c.f(W)$ = false **then**
   $B$=$H.f_B(B, W, C)$
   **endif**
   **endfor**
   **for all** $p \in L.P_L$ **do**
   **if** $p.f(W)$ **then**
   $T.T_C := T.T_C \cup$ new $T\{$ new $C\{W, P\}, \{\}, B.L, T\}$
   **goto** 2.
   **endif**
   **endfor**
   **if** $E_n \neq \{\}$ **goto** 3. **Endif**
   **if** $g.f(W)$ $\forall g \in W.G_L$ return plan **endif**
4. Stalling has occurred. $B$:=$H.f_B(T, W, StallC)$ **goto** 2.

Fig. 6. The Planning Branch algorithm performs a state-space search incorporating domain-dependent knowledge.

The World Model $W$ contains the invariant portion of the problem. This includes the Mathematica definition for the chassis and jobs. Conversely, the State contains the variable portion of the problem including the current schedule, $T_{AC}$, and $T_i$.

For this domain, it is sufficient to trigger a decision epoch each time the Effects of the last planning step have successfully been applied.

Because Checkpoints are constructed by the planning algorithm a Checkpoint Factory is required to enable dependency injection for domain-specific Checkpoint creation. The Checkpoint contains job slack times with respect to the existing schedule and an iterator which creates the next schedule for the heuristic.

The majority of the planning logic occurs in the Planning Heuristic. This is where most of the ambiguity of how to solve the problem resides and where this work differs significantly from domain-configurable planners. The Planning Branch algorithm makes no attempts to guarantee completeness or optimality except for what the domain experts guarantee within their implementation. However, there is nothing preventing a generalized heuristic from working within this algorithm which solves general planning problems for constrained models exactly the same way academic planning solutions do today. The problem could even be defined in an existing planning language. Hence, this work is compatible with other existing research techniques in planning.

The planning heuristic encodes many pieces of domain-specific knowledge which it uses to solve this domain efficiently. This includes:

- A basic thermal-ignorant earliest deadline first (EDF) scheduler.
- A slack evaluator which determines time ranges for each chassis a job could be migrated to.
- An overcooling evaluator which determines how much heat could be added to each chassis at each point in time without modifying the current $T_{AC}$ function.
- A solver for a $T_{AC}$ function which does not violate the thermal constraints. Our implementation uses simple piecewise-linear segments such as those shown in Figure 4.
- A job migration function which migrates jobs from areas of overcooling to more optimal areas.
- A job migration function which migrates sets of jobs from thermal peaks to thermal valleys while improving overall energy.

The result of all of this is a greedy, but sub-polynomial, solution which finds a sort of local optimum of a job schedule and $T_{AC}$ function. As the schedule slack diminishes the thermal-aware schedule approaches that of a thermal-ignorant schedule. As the schedule slack increases jobs will be spread out in time and across chassis such that waste cooling and thermal peaks are minimized. This capitalizes directly on our intuition about the domain and all parts of this can be refined until a solution is reached which provides an acceptable tradeoff of runtime vs energy efficiency.

# IV. Related Work

The semantic mapping issues identified by our example domain were caused by non-Markovian actions and discretizing continuous variables.

Work has been done on the former using discretized temporal logic and on non-Markovian rewards but no work we are aware of solves non-Markovian models in a continuous space. [14-16]

Some work has also been done identifying the problems caused by discretizing continuous representations in spatial domains and countering these challenges using domain predictive control methods. [17]

Older work exists on PDDL+ on hybrid domains with a combination of discrete and continuous signals. [18] A AAAI workshop is scheduled to address the challenges of these hybrid domains in August which should greatly expand the existing work on this topic.

Facilitating domain author control within the planning process is something that has fallen out of fashion despite its early successes. [5] Over a decade ago, work which capitalized on domain-specific knowledge was more common. In the 2000 and 2002 International Planning Competitions domain-configurable planners dominated the performance with planners like TLPlan, TALPlanner, and SHOP2. [19-21] Authors of this early work believed that this direction was not only beneficial but indispensable.

> "… can often convert an intractable planning problem to a tractable one; i.e., it can often be the only way in which automatic planning is possible." [19]

> "The approach appears to be the key to the next order of magnitude scaling of state-space planning algorithms. Problems that could not be solved in days could be solved in seconds with additional axiomatic knowledge." [22]

Since then no domain-configurable heuristic languages have been officially developed. Rather, domain-configurable planners exist with no unifying representation. The lack of progress and apparent regression of domain-configurable work gives the general appearance of a cultural bias in planning against domain-configurable work.

As further evidence, most recent work in this direction has appeared immediately outside the planning community in areas like logical programming and answer set programming. [23-24] All examples of domain-configurable work we are aware of impose modeling limitations. This is an unfortunate state of affairs for CPS owners and leads many applications to use simpler, less efficient technologies.

> "Given the hardness of generating domain models for planning, many users are not exploiting automated planning but use easier approaches, even if they are less efficient [25]."

Despite this, work on cyber-physical systems continues to grow. Even within the narrow context of planning there have already been many notable scientific achievements. Some high-profile examples include satellites and the ISS, the Mars rovers, and the DARPA grand challenge winner. [26-28]

However, for the foreseeable future, physical models will exist on or beyond the frontier of expressiveness. As long as this remains the case we recommend that more thought be given not only to the components as they stand but also how they can be extended to models of arbitrary complexity.

# V. Conclusions

Accurate models are important because they are necessary to ensure the safety of the system and improve efficiency by cutting slack from the margin of error. Extensible components are important because they enable accurate models to be used in practice. This can improve performance and allow refinement when a component isn't sufficient for the task. Both of these empower CPS developers to produce more refined and impactful systems. We as a community should develop extensible components because ultimately they have a greater impact on the bottom line.

# References


[1] Laird, J. (2012). The Soar cognitive architecture. MIT Press.

[2] Anderson, J. R. (2013). The architecture of cognition. Psychology Press.

[3] Langley, P. (2006). Cognitive architectures and general intelligent systems.AI magazine, 27(2), 33.

[4] Kambhampati, S. (1997). Refinement planning as a unifying framework for plan synthesis. AI magazine, 18(2), 67.

[5] Nau, D. S. (2007). Current trends in automated planning. AI magazine, 28(4), 43.

[6] Clarke, E. M., Grumberg, O., & Peled, D. A. (1999). Model Checking The MIT Press. Cambridge, Massachusetts, London, UK.

[7] Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., & Gupta, A. (2013). Probabilistic temporal logic falsification of cyber-physical systems. ACM Transactions on Embedded Computing Systems (TECS), 12(2s), 95.

[8] Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., ... & Sun, Y. (1998). PDDL| The Planning Domain Definition Language.

[9] Pednault, E. P. (1989, December). ADL: Exploring the middle ground between STRIPS and the situation calculus. In Proceedings of the first international conference on Principles of knowledge representation and reasoning (pp. 324-332). Morgan Kaufmann Publishers Inc..

[10] Jonas, M., Gilbert, R. R., Ferguson, J., Varsamopoulos, G., & Gupta, S. K. (2012, June). A transient model for data center thermal prediction. In *Green Computing Conference (IGCC), 2012 International* (pp. 1-10). IEEE.

[11] Moore, J. D., Chase, J. S., Ranganathan, P., & Sharma, R. K. (2005, April). Making Scheduling" Cool": Temperature-Aware Workload Placement in Data Centers. In USENIX annual technical conference, General Track (pp. 61-75).

[12] Fox, M., & Long, D. (2002). PDDL+: Modelling continuous time-dependent effects. In Proc. 3rd International NASA Workshop on Planning and Scheduling for Space.

[13] Jonas, M. (2011). JPDL: A fresh approach to planning domain modeling. *KEPS 2011*, 63.

[14] Thiébaux, S., Gretton, C., Slaney, J. K., Price, D., & Kabanza, F. (2006). Decision-Theoretic Planning with non-Markovian Rewards. *J. Artif. Intell. Res.(JAIR)*, *25*, 17-74

[15] Gabaldon, A. (2002, July). Non-markovian control in the situation calculus. In*AAAI/IAAI* (pp. 519-525).

[16] Gonzalez, G., Baral, C., & Gelfond, M. (2003, August). Alan: An action language for non-markovian domains. In *NonMon. Reasoning, Action and Change Workshop*.

[17] Löhr, J., Wehrle, M., Fox, M., & Nebel, B. (2014, June). Symbolic Domain Predictive Control. In *Proceedings of the 28th National Conference on Artificial Intelligence (AAAI 2014)*.

[18] Fox, M., & Long, D. (2006). Modelling Mixed Discrete-Continuous Domains for Planning. J. Artif. Intell. Res.(JAIR), 27, 235-297.

[19] Bacchus, F., & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. Artificial Intelligence, 116(1), 123-191.

[20] Kvarnström, J., & Doherty, P. (2000). TALplanner: A temporal logic based forward chaining planner. Annals of Mathematics and Artificial Intelligence,30(1-4), 119-169.

[21] Nau, D. S., Au, T. C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. J. Artif. Intell. Res.(JAIR), 20, 379-404.

[22] Kautz, H. (1998). The Role of Domain-Specific Knowledge in the Planning as Satisfiability Framework.

[23] Zhou, N. F., Bartak, R., & Dovier, A. (2015). Planning as tabled logic programming. Theory and Practice of Logic Programming, 15(4-5), 543-558.

[24] Gebser, M., Kaufmann, B., Romero, J., Otero, R., Schaub, T., & Wanko, P. (2013, July). Domain-Specific Heuristics in Answer Set Programming. InAAAI.

[25] Shah, M., Chrpa, L., Jimoh, F., Kitchin, D., McCluskey, T., Parkinson, S., & Vallati, M. (2013). Knowledge engineering tools in planning: State-of-the-art and future challenges. Knowledge Engineering for Planning and Scheduling, 53

[26] Muscettola, N., Nayak, P. P., Pell, B., & Williams, B. C. (1998). Remote agent: To boldly go where no AI system has gone before. Artificial Intelligence, 103(1), 5-47.

[27] Estlin, T., Castano, R., Anderson, R., Gaines, D., Fisher, F., & Judd, M. (2003, August). Learning and planning for mars rover science. In Proc. IJCAI Workshop on Issues in Designing Physical Agents for Dynamic Real Time Environments: World Modelling, Planning, Learning and Communicating.

[28] Ventures, M. D. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of field Robotics*, *23*(9), 661-692.

# Embedded CPS for Real-Time Monitoring of a Laser Beam Deflection System Using Spectral Analysis

**B. Arejita**[1,2], **M. Antunez**[1], **J. Diaz**[1], **A. Ochoa**[1]
[1]Ikergune A.I.E., San Antolín 3, 20870 Elgoibar, Spain.
**barejita@ikergune.com**, mantunez@ikergune.com, jdiaz@ikergune.com, aochoa@ikergune.com
[2]Department of Telecommunications and Electronics, UPV/EHU, Bilbao, Spain.

**Abstract –** *Cyber-Physical Systems (CPS) are seen as true technology enablers for new complex industrial applications from the perspective of the Industry 4.0. In this context, the challenges of advanced laser material processing applications present an interesting research area where CPS could be used for the real-time control and monitoring of such complex processes. In this article we present a method to monitor the correctness of the commanded position of galvanometer scanners using real time spectral analysis. We also provide a reference design based on FPGA that tests the proposed method. We also present a case study that has proven to be effective in the detection of high frequency components and target laser pattern degradation.*

**Keywords:** CPS, Laser Manufacturing, Galvanometer scanner, FFT, FPGA

## 1 Introduction

In the last couple of years, initiatives such as Industry 4.0 in Europe or industrial internet and the advanced manufacturing in the USA are experiencing an important momentum [1]. The idea of integrating ICT technologies in the industrial value chain has been an important research topic for years, but it is now when the concept of a new industrial revolution based on smart transducer, the power of the cloud and Industrial Internet of Things (IIoT) devices and their integration with manufacturing management systems is getting the support from governments, policy makers and companies [2].

On the other hand, data privacy and cybersecurity issues have been some of the most important factors that have slowed down the introduction of ICT in industrial environments. There has also been a discussion in the research community whether these new concepts are real and have come to stay or if they just are empty concepts that have been boosted by the hype [3].

Nevertheless, due to the Moore's law, the computational power of new generation integrated circuits has improved in such a way that it is now possible to implement complex cryptographic algorithms and real time control and communication applications in embedded systems. Companies like Xilinx are making a huge effort developing new FPGA architectures such as the new Ultrascale MPSoC to solve the computational and connectivity needs of these new generation industrial embedded devices [4]. In addition, communication security algorithms have reached a point of maturity in a way that important industrial stakeholders can rely on the technology.

In this regard, Cyber Physical Systems (CPS) play an important role in this upcoming industrial revolution. CPS systems are connected computational nodes that interact with their surrounding physical world and processes enabling data access and data processing services through the internet with a special focus on data integrity, data reliability, privacy and communication security [5].

The introduction of CPS elements in industrial environments has coined the term Cyber-Physical Production System (CPPS). Such devices will potentially allow the migration from the traditional automation pyramid towards a more distributed architecture, which will blur the boundaries between process control, plant management and enterprise and resource management levels.

In order to secure the adoption of the concepts embraced by the Industry 4.0, CPPS systems will have to address several R&D challenges:

- *Predictability and Adaptability:* using reconfigurable manufacturing systems (RMS) [6].
- *Interoperability:* with the introduction of high level communication protocols such as OPC-UA[7].
- *Communication security:* with the help of standards such as ISA/IEC-62443 for industrial automation and control communication security [8].
- *Robustness and scalability:* through the usage of cloud computing and resource virtualization. [9]
- *Autonomy and cooperation:* with the implementation of holonic architectures [10].

CPPS can thus be seen as technology enablers for complex industrial processes. In this regards, advanced laser material processing applications present an interesting topic where the capabilities of CPPS systems could be used in the monitoring and control of the laser process [11]. In this work we focus on laser applications where laser galvanometer scanners are used to deflect the laser beam and generate laser patterns in a way that a controlled energy deposition on the material is achieved.

The paper is organized as follows: section 2 is dedicated to describe the motivation of this work and makes a brief overview of laser beam deflection systems. In section 3, we present our proposed solution for the real-time monitoring

system of the galvanometer scanner positioning commands based on frequency analysis. Section 4, describes the implementation of the proposed solution in a Xilinx SoC. In section 5 a case study is analyzed focusing on the generation of a simple laser beam pattern. Sections 6 presents a brief discussion on the obtained results and future work and finally in section 7 we present our conclusions summarizing the work that has been carried out.

## 2 Motivation and state of the art

The main element of a laser beam deflection system is a 2D galvanometer scanner. A 2D galvanometer scanner is composed of two mirrors coupled into their respective galvanometers which are precisely positioned to deflect the incident laser beam in the X and Y direction. Each galvanometer is a mechatronic system with three main components: an actuator, typically a permanent moving magnet DC motor, a detector for the position of the mirror, and an electronic drive that produces control signals to position the mirror in the desired angle [13][14].

The deflected laser beam is then focused into the image plane using an f-theta lens. In ordinary lenses the laser beam position depends on the tangent of the scan angle whereas f-theta lenses present a linear relationship between position on the image plane and the scan angle. In addition to this the typical scanning angles are small enough so that the relationship $\theta \approx \tan(\theta)$ can be considered to be true [15]. This yields to the following relationship between the scan angle and the position in the image plane:

$$x = f \times \tan\theta_x \approx f \times \theta_x$$
$$y = f \times \tan\theta_y \approx f \times \theta_y \quad (1)$$

where f is the focal length of the f-theta lens and $\theta_x$ and $\theta_y$ the scan angle for x and y axis respectively. Figure 1 shows the diagram of a 2D galvanometer scanner.

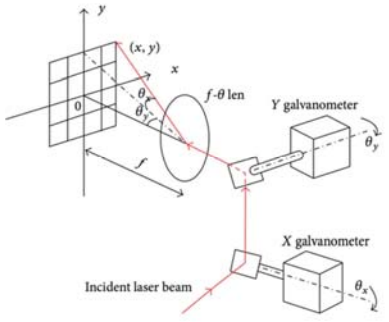



Figure 1. Laser beam deflection process using X-Y galvanometer system [15].

Most commercial galvanometer scanner systems use a common communication protocol called XY2-100. The protocol provides a serial communication for each laser beam deflection axis and common clock and synchronization signals for the serial communication streams generated by the master. The maximum clock speed is usually limited by the capabilities of the deflection system and the galvanometer position controller electronics, but it is typical to find a 2 MHz clock signal in most of the systems. The sync signal asserts a logical cero at the last bit of each data frame otherwise it remains at logical one, and it is used to notify the end of frame to the slave. Each data frame consists of 20 bits of length which yields to a 10 µs frame duration with a 2 MHz clock signal. Each data frame is composed of three fields: Control, data and parity. The three most significant bits are dedicated to the control command, the following 16 bits compose the data field and the least significant bit is a parity bit with even parity. Figure 2 shows the XY2-100 communication protocol diagram.

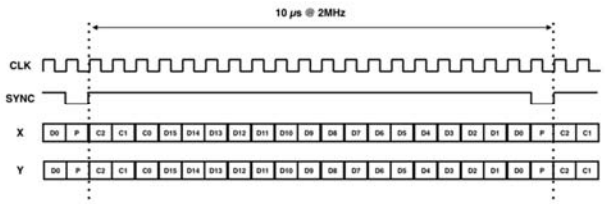



Figure 2. XY2-100 galvanometer scanner position control protocol.

It is out of the scope of this paper to present a complete description of the protocol and just the basic control command is going to be analyzed; the set position offset command. The set position offset command is used to position the galvanometer within its value range (control command: 0x03 hex). The position set point value is specified using the 2 Byte data field providing a $2^{16}$ step resolution.

The spatial resolution in the image field directly depends on the distance between the image plane and the f-theta lens and the maximum deflection angle achievable by the galvanometer scanner. In some applications where the spatial resolution is not important a greater distance can be used between the deflection unit and the image plane, obtaining higher speeds as the same pattern can be drawn with lower angles.

The frequency response of a galvanometer scanner system dramatically decays for frequencies above 1kHz and it is not able to respond to frequencies above 5kHz [14]. As a consequence, when a galvanometer scanner is exposed to these frequency ranges the servo actuator works in an overloaded working zone, experiencing high mechanical stresses that will cause the actuator wearing and eventually yielding to a system failure. This rises the need of a protection system against high frequency position commands. In addition to this, there are several advanced material processing applications in which both the energy addition kinematics and patterning of the laser beam are a key factor of the process [16][17].

Most galvanometer scanner controller cards run under non-deterministic operating systems which can introduce errors to the commanded position. In addition to this, the commands sent to a galvanometer controller could be wrong due to a software bug or an operating system crush. This and

other reasons raise the need of a monitoring system to activate alarms and protect the laser beam deflection unit from high frequency position modifications and monitor the correctness of the commanded patterns.

## 3 Proposed solution

In this work we propose a real time monitoring method for 2D galvanometer scanner position control commands based on the spectral analysis of the commanded patterns using short time Fourier transforms.

Every band limited signal can be reconstructed from its frequency transform when the sampling rate is greater than two times the maximum frequency of the signal [18]. In this regard, the frequency spectrum of a band limited signal could be used to characterize it.

The XY2-100 protocol provides 100k samples per second at at typical clock rate of 2 MHz, and the commanded patterns will rarely be grater than 1 kHz due to the physical limitations of a galvanometer. The commanded position signal can be modeled as an oversampled discrete signal of the figure to be generated. As the maximum frequency to be monitored is 5 kHz, the signal must be decimated by a factor of 10 to obtain an equivalent sampling frequency of 10 kS/s. In order to avoid aliasing, an antialiasing filter must be applied before decimation [18].

$$x_M[n] = x_f[nM] \tag{2}$$

where $x_f[n]$ is the pattern signal sampled at 100 kS/s and filtered by the antialiasing filter with $\omega_c = \pi/M$ and $x_M[n]$ is the decimated signal. The samples are then stored in a buffer to feed the Discrete Fourier Transform (DFT).

Before applying the DFT, a windowing function is multiplied to the buffered samples to minimize the effects of discontinuities that occur when there is not a coherent sampling [19][7]. The resulting signal is then fed to the DFT computing the Short Time Fourier Transform (STFT) with an overlap of O and L samples between adjacent blocks.

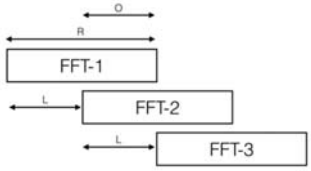



Figure 3. STFT: sample overlapping scheme between consecutive FFT computations.

The short time Fourier transform of the decimated signal is mathematically defined as:

$$\begin{aligned} X(\omega, Lm) &= DTFT(x_M[n - Lm] \cdot w[n]) \\ &= \sum_{n=-\infty}^{\infty} x_M[n - Lm] \cdot w[n] \cdot e^{-i\omega n} \\ &= \sum_{n=0}^{R-1} x_M[n - Lm] \cdot w[n] \cdot e^{-i\omega n} \end{aligned} \tag{3}$$

were DTFT is the Discrete Time Fourier Transform. In order to have a discrete STFT, the continuous STFT is sampled in the frequency domain from ω=0 to ω=2π.

$$\forall\, k, 0 \le k \le NFFT - 1 : (\omega_k = \frac{2\pi}{NFFT} k)$$

The discrete STFT is then expressed as:

$$\begin{aligned} X_D[k, Lm] &= X(\frac{2\pi}{NFFT} k, Lm) \\ &= \sum_{n=0}^{R-1} x_M[n - Lm] \cdot w[n] \cdot e^{-i\frac{2\pi k}{NFFT} n} \\ &= \sum_{n=0}^{R-1} x_M[n - Lm] \cdot w[n] \cdot {W_{NFFT}}^{-kn} \end{aligned} \tag{4}$$

The DFT is calculated using a Fast Fourier Transform (FFT), and the result compared to the pattern FFT. The difference signal is finally evaluated to detect any perturbation with respect to the pattern figure.

Figure 4 shows the diagram of the proposed galvanometer scanner position command monitoring system for one channel.

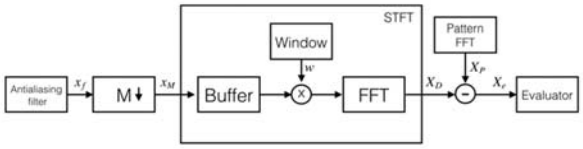



Figure 4. Galvanometer scanner position monitoring system.

The evaluator is able to detect malfunctioning situations by analyzing the energy in each frequency bin and comparing it to the pattern frequency domain signal.

## 4 Implementation

In order to test the proposed method, we have used a RAIO laser embedded control platform. The RAIO platform is based on a Xilinx Zynq System on Chip (SoC) which incorporates a dual core ARM Cortex-A9 based processing system (PS) and a Xilinx programmable logic (PL) in a single device and multiple XY2-100 and laser control interfaces. The PL allows to program the XY2-100 and the signal processing algorithms in hardware, so that a real time monitoring system can be implemented.

For the FFT a comparative study has been done evaluating different radix-4 and radix-2 implementations in

hardware and software for several FFT sizes as shown in Figure 5. The hardware implementations of the FFT have been tested using Xilinx's FFT IP-core running at 200 MHz. The multiply and accumulate operations have been performed in the DSPs of the FPGA and the core has been configured to be a dual channel FFT so that X and Y channels can be computed in parallel. The input data are complex values expressed in 32 bit floating point format and as the input samples are real values the imaginary parts are zero valued.

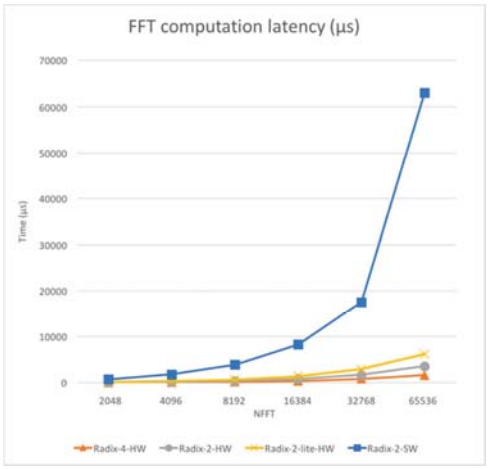



Figure 5. FFT computation latency comparison.

As the sampling frequency is 10 kS/s the maximum frequency to be monitored is 5 kHz, which is more than enough for the galvanometer scanner position signal. In order to obtain a frequency resolution of at least 0.5 Hz an FFT of 32768 points must be used, obtaining an effective frequency resolution of 0.3 Hz. Using the radix-4 algorithm in HW takes 2 ms for both X and Y channels. The time to fill each FFT buffer at the specified sample rate is 3.27 seconds and in order to have a sub-second time resolution sample overlapping must be applied. Table 1 shows different achievable time resolutions for the STFT for different overlapping values.

Table 1. Time shift between FFT computations depending on the overlapping amount.

| | Overlap | | | |
|---|---|---|---|---|
| | 50% | 75% | 90% | 95% |
| Time (s) | 1,6384 | 0,8192 | 0,3277 | 0,1638 |

Taking all this information into account the final design has been done with a 32768-point radix-4 algorithm implemented in hardware using an overlap of 90% between simultaneous FFTs.

For the synchronization and communication with external components the HSR/PRP switch and IEEE1588 IP-cores from SoC-e have been used. The HSR/PRP IP-core provides high availability and communication redundancy using two networks and duplicating the frames. The duplicated frames must then be analyzed and discarded but the IP-core makes all this processing transparently. HSR works similarly but using a single LAN being the basic topology a ring [20]. The HSR/PRP switch is used in the design to communicate with the other elements in the network which also use HSR/PRP. It is also important to synchronize all the elements in the network and the PTP protocol has been selected for that purpose. The PTP protocol as defined in IEEE1588-2008 standard provides synchronization accuracies in the nanosecond range [21]. PTP has also been used to provide the time stamping of the alarms using a common time base.

Figure 6 shows the diagram of the SoC design. The XY2-100 protocol signals are received through the XY2-100 IN interface and are redirected to the XY2-100 OUT interface allowing analyzing the XY2-100 protocol without interfering the process. The received points are then transmitted to the evaluator using an AXI4 stream channel to feed the FFT core in the evaluator. This allows performing an on-the-fly data analysis exclusively using programmable logic components to raise alarms through a General Purpose Input Output (GPIO) interface. The evaluator IP-core can be configured and monitored using an AXI4 interface to communicate it with the processor system (PS). Finally, the IEEE 1588 and HSR/PRP cores are connected to the PS to provide precise timing and high availability and redundancy capabilities to the system.

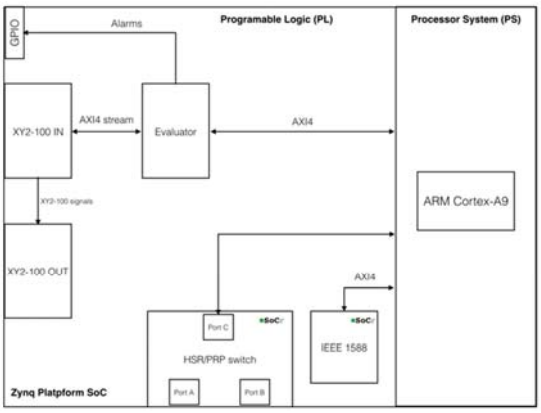



Figure 6. Implementation of the design in a Xilinx Zynq SoC.

## 5 Case study: Elliptic pattern

To test the presented design a simple elliptic pattern has been chosen as a reference. The normalized X and Y axes signals are presented bellow:

$$x = 0.5 + 0.1\cos(2\pi \cdot 500 \cdot t)$$
$$y = 0.5 + 0.2\sin(2\pi \cdot 500 \cdot t) \quad (5)$$

Figure 7 shows the received signal after decimation resulting in a discrete signal sampled at 10 kS/s. The X and Y signals commanded to the galvanometer scanner generate the elliptic pattern.

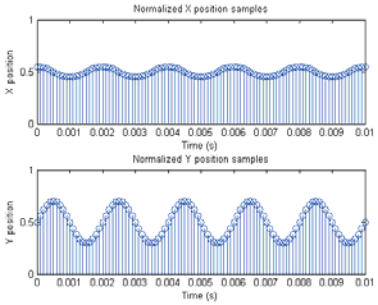



Figure 7. 5 periods of normalized X and Y position command sampled signals (10 kS/s).

The frequency domain representation of this signals can be used to monitor the received pattern. Figure 8 shows the FFT of X and Y signals module and phase. The signals have been windowed using a Kaiser window and fed to the 32768 point FFT.

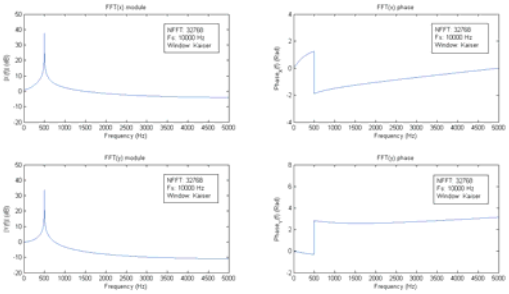


Figure 8. Frequency domain characterization of the elliptic laser beam pattern (NFFT: 32768, Sampling frequency: 10 kS/s and Kaiser window).

To test the system, fault injection has been carried out in channel X. For the X signal after 2 seconds a frequency shift has been introduced for 2 seconds, which simulates a deformation in the pattern. In a real situation, this would be equivalent to a wrong laser energy addition to the material that could result in a wrong material processing. Then after 6 seconds from the beginning of the test high frequency components are added to the signal. High frequencies put the galvanometer scanner under high mechanical stresses that can lead to mechanical degradation and eventually to a system failure. Both of this situations must be detected in order to prevent galvanometer degradation and process degradation. Figure 9 shows the spectrogram of the pattern signal, the commanded signal and the difference of the spectrograms. The spectrogram shows how the frequency spectrum of the signal changes over time. Comparing the measured signals frequency to the pattern it is possible to detect undesirable frequency components that can signal malfunctioning situations.

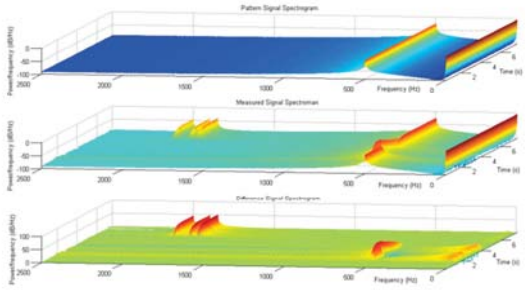


Figure 9. (a) Pattern signal spectrogram, (b) Measured signal spectrogram with fault injection, (c) Difference signal spectrogram fed to the evaluator.

The evaluator is able to detect undesired high frequencies and pattern degradation situations by analyzing the energy in each frequency bin. As the frequency resolution is 0.3 Hz and time resolution is 32 ms, the system is able to accurately detect and signal the external components (laser power generator and laser scan command generator) protecting both the galvanometers and the processed material.

## 6 Discussion

The presented solution has shown to be strong detecting errors in the laser scanner position commands. It is especially efficient at detecting high frequencies that can deteriorate the mechanical parts of the galvanometer scanner. In the case of simple patterns, the detection of signal degradation has also shown to be efficient, being able to raise an alarm and stop the laser process in 32.7 ms.

In the case of complex patterns, the detection of changes in the frequency spectrum of the signal results to be more challenging. Even though, changes in the signal spectrum are easily detectable, it is more difficult to relate spectrum changes to a specific cause. This complicates the control strategy to be followed by the laser power and scanning controller.

For complex patterns it can be interesting to use machine learning techniques and classifier algorithms to help relating frequency modifications to specific causes so that a proper laser control strategy can be carried out.

Some commercial galvanometer scanners also provide position feedback for each channel. In this regard, future work will be focused on studying the cross-correlation of the commanded and feedback signal that can help to accurately detect the galvanometer scanner angular drift so that a real-time corrected position can be commanded to each galvanometer.

A lower frequency resolution could also be used for the same application making it possible to implement the FFT in SW. Nevertheless, it has to be taken into account that the FFT implementation in hardware allows to compute the X and Y transforms in parallel. Moreover, when computed in hardware the CPU remains idle and the processor time can be used for other purposes or go to a CPU sleep mode obtaining a very

energy efficient implementation. In addition to this, the configurable nature of the FPGA can allow to monitor different laser scan heads in parallel which in software would have been impossible to achieve with a dual core ARM Cortex-A9.

# 7 Conclusions

CPPS systems are emerging as the distributed cooperative entities that will enable the next industrial revolution bringing a more efficient industrial environment with a higher level of automation improving the robustness, adaptability and predictability of industrial manufacturing processes with a significant increase in productivity.

In this context, we have presented a solution to automatize the monitoring of pattern based laser beam deflection systems treating the laser position command signals as periodic discrete signals that can be characterized and evaluated in the frequency domain.

We have then presented a reference design that implements the proposed method in a Xilinx based System on Chip, implementing the signal processing directly in the FPGA part of the device. In order to have robustness and redundancy in the communications we have introduced an HSR/PRP IP-core in the design and a PTP IP-core for the synchronization of the elements of the network and sub-microsecond time stamping for the alarm events.

With the presented solution it is possible to automatically detect high frequency components in the galvanometer commanded position and deviations in the commanded patterns with a resolution in frequency of 0.3 Hz and a resolution in time of 32 ms. A better time resolution could be achieved by using a higher overlapping value between FFT iterations but this would introduce unwanted effects in the frequency domain. Another solution would be to decrease the amount of decimation in a way that a higher effective sample rate would be achieved, allowing to fill faster the input sample buffer. This solution would need an FFT of more samples in order to get an equivalent frequency resolution.

The detection of high frequency components results to be very effective allowing to signal the scan controller so that the damage to the scan head can be minimized. The detection of the degradation of simple patterns can also be very effective monitoring pattern frequency shifts and pattern degradations.

The proposed method can also be used to make an estimation of the galvanometer scanner angle drift in time by comparing the commanded and actual position when special galvanometer scanners with position feedback are used. Future work will be focused on the study of the correlation of the commanded and actual galvanometer scanner position, allowing to infer galvanometer scanner angle drifts in real time allowing to adjust the commanded position taking into account the instantaneous angle drift. Future work will also cover a more robust method to monitor complex laser patterns.

# 8 Acknowledgement

We would like to thank Talens/Etxe-tar group (www.talenssys.com), for providing the RAIO laser quality assurance and control embedded platform and for its strong support to Industry 4.0 and CPPS solutions for advanced industrial laser manufacturing processes. We would also like to thank SoC-e for providing their HSR/PRP switch and IEEE1588 IP-cores and their strong support to developments of secure CPPS systems (www.iiconsortium.org/security-claim). Last but not least we would like to recognize the effort that Xilinx is doing to provide hardware and software platforms for the development of high added value solutions.

# 9 References

[1] A. Ochoa, J. Diaz, B. Kamp. "Smartization of products and buyer-supplier relationships as enablers for servitization: evidence from a machine tool manufacturer in the context of smart manufacturing". 4th International Business Servitization Conference, Nov 2015.

[2] C. Toro, I. Barandiaran, J. Posada. "A Perspective on Knowledge Based and Intelligent Systems Implementation in Industrie 4.0". Procedia Computer Science (Elsevier), Vol. 20, pp. 362-370, 2015.

[3] R. Drath, A. Horch."Industrie 4.0: Hit or Hype?". IEEE Industrial Electronics Magazine (IEEE), Vol. 8, issue 2, pp. 56-58, Jun 2014.

[4] Xilinx. "Xilinx Ultrascale MPSoC Architecture". Backgrounder, online publication (Xilinx), http://goo.gl/OfPSoq, Mar 2016.

[5] L. Monostori. "Cyber-Physical Production Systems: Roots, Expectations and R&D Challenges". Procedings of the 47th CIRP Conference on Manufacturing Systems, Variety Management in Manufacturing (Elsevier), Vol. 17, pp. 9-13, 2014.

[6] Y.Koren, Z. Heisel, F. Jovane, M. Moriwaki, G. Pritschow, G. Ulsoy, H. Van Brussel. "Reconfigurable Manufacturing Systems". CIRP annals, Manufacturing technology, Vol 48, issue 2, pp. 527-540, 1999.

[7] W. Whanke, S. Leither, M. Damm, "OPC Unified Automation", Springer, 2009.

[8] W. Knowles, D. Prince, D. Hutchison, J. F. Pagna Disso, K. Jones. "A survey of cyber security management in industrial control systems". International Journal of Critical Infrastructure Protection (Elsevier), Vol. 9, pp. 52-80, Jun 2015.

[9] R.Piggin. "Are industrial control systems ready for the cloud?", International Journal of Critical Infrastructure Protection (Elsevier), Vol. 9, pp.38-40, Jun 2015.

[10] J. Barbosa, P. Leitao, E. Adam, D. Trentesaux. "Dynamic self-organization in holonic multi-agent manufacturing systems: the ADACOR evolution ". Computers in Industry (Elsevier), Vol. 66, pp. 99-111, Jan 2015.

[11] J. Diaz, C. Bielza, J. L. Ocaña, P. Larrañaga. "Development of a Cyber-Physical System based on selective Gaussian naïve Bayes model for a self-predict laser surface heat treatment proess control". Machine learning for Cyber Physical Systems ML4CPS (Springer Vieweg), 2015.

[12] Han Woong Yoo, S. Ito, M. Verhaegen, G. Schitter. "Transformation-based Iterative Learning Control for Non-collocated Sensing of a Galvanometer Scanner". 2013 European Control Conference (ECC), pp. 1204-1209, 2013.

[13] G. F. Marshall, Ed. "Handbook of Optical and Laser Scanning". Marcell Decker, 2004.

[14] C. A. Mnerie, S. Preitl, V. F. Duma. "Performance Enhancement of Galvanometer Scanners Using Extended Control Structures". 8th IEEE Symposium on Applied Computational Intelligence and informatics, pp. 127-130, May 2013.

[15] W. Dongyun, Y. Xinpiao, "An Embedded Laser Marking Controller based on ARM and FPGA Processors". The scientific world journal (Hindawi Publishing Corporation), Vol. 2014, March 2014.

[16] H.J. Booth. "Recent Applications of Pulsed Lasers in Advanced Material Processing". Proceedings of Symposium H on Photonic Processing of Surfaces, Thin Films and Devices, of the E-MRS2003 Srping Conference (Elsevier), Vol. 453-454, pp. 450-457, Apr 2004.

[17] F. Cordovilla, A. Garcia-Beltran, J. Dominguez, P Sancho, J. L. Ocaña. "Numerical-experimental analysis of the effect of surface oxidation on the laser transformation hardening Cr-Mo steels". Applied Surface Science (Elsevier), Vol. 357, pp. 1236-1243, Dec 2015.

[18] A. V. Oppenheim, R. W. Schafer, J R. Buck. "Discrete-Time Signal Processing". Prentice Hall, 1999.

[19] F. J. Harris. "On the use of windows for harmonic analysis with the discrete Fourier transform". Proceedings of the IEEE (IEEE), Vol.66, Issue 1, pp. 51-83, Jan 1978.

[20] J. A. Araujo, J. Lazaro, A. Astarloa, A.Zuloaga, J. I. Gárate. "PRP and HSR for High Availability Networks in Power Utility Automation: A Method for Redundant Frames Discarding". IEE transactions on Smart Grid (IEEE), Vol.6, issue 5, pp. 2325-2332, Jan 2015.

[21] N.Moreira, J. Lazaro, U. Bidarte, J. Jimenez, A. Astarloa. "On the Utilization of System-on-Chip Platforms to Achieve Nanosecond Synchronization Accuracies in Substation Automation Systems". IEEE transactions on Smart Grid (IEEE), Vol. PP, issue 99, pp. 1-11, Jan 2016.

ISBN: 1-60132-433-2, CSREA Press ©

# SESSION

# LATE BREAKING PAPERS

## Chair(s)

**TBA**

ISBN: 1-60132-433-2, CSREA Press ©

ISBN: 1-60132-433-2, CSREA Press ©

# Using Head Mounted Display to Increase Field-of-View for Futuristic User Interface

**Chandler Johnson, Sudip Chakraborty**
Department of Computer Science, Valdosta State University, Valdosta, GA, USA

**Abstract** – *User Interface (UI) is one of the primary components of any operating system. In many cases, efficiency or ease in using a feature provided by the operating system depends on the UI. Advent of touchscreen technology makes this issue more prevalent in mobile devices like smartphones and tablet computers. In this paper we investigate four user interfaces in their vanilla state, including both mobile and desktop platforms, and identify the limitations the UI imposes on interaction with the system. We suggest that increasing user's field-of-view (FOV) can mitigate several such limitations. In particular, we explore the use of head mounted display (HMD) over traditional two-dimensional screen to increase user's FOV. Finally, we use mathematical analysis to substantiate our claim. This paper is our attempt to propose a direction in the future of user interfaces of operating systems.*

**Keywords:** User Interface, Head Mounted Display, Field-of-View, Operating Systems

## 1 Introduction

The manyfold increase in popularity of personal computers has forced the computer manufacturers to ensure that the demands of the user base are met. One of the primary demands of the users is usability of the computing system. Especially, the users demand systems to be easier to interact with, that is, more intuitive and easy to handle user interface (UI). Therefore, developing new user interface or supporting newer interaction mechanisms in existing operating systems is a constant challenge for system designers. In the past, the design of the operating system was limited by the existing hardware. Consequently, users of a computing system had to interface directly with hardware as the concept of operating system was at primitive stage. With the introduction of video display terminals (VDT), operating systems evolved to add more features and increase usability of the systems. This evolution led to advent of command-line interfaces (CLI) that became one of the primary mechanisms used for UI. However, it was not until the first personal computer with actual graphical interface supporting colors that the usability of computers increased. Since then UI designs have come a long way but still primarily focused on supporting system features in a two-dimensional way. With increasing size of monitors and support for multiple monitors users now can utilize the UI more efficiently, but are limited to restricted field-of-view supported by two-dimensional screens. Mobile devices and touchscreen technologies have added extra dimension to the UI design challenges. Some standard functionality, like navigating the file system, provided by an operating system largely depends on the UI support. Also, modern users are keen to move toward intuitive ways (like voice control, or gesture control) to interact with a system. This trend is clearly evident from growing use of devices (e.g., head mounted displays) supporting virtual reality.

In this paper we suggest use of such devices to address the limitations that current UIs are facing. In particular we propose use of HMDs, rather than a typical 2D monitor, to increase a user's filed-of-view. To justify our suggestion we examine four major operating systems and highlight their limitations to provide some features (we have selected file system navigation) due to current state of the UIs. Though there are many modifications available to each of these operating systems, for consistency we refer to the *vanilla state* of the UI after a fresh install. We show mathematically how the FOV can be enhanced using an HMD and thereby mitigates the current UI limitations. This is our attempt to provide a futuristic solution to UI limitation.

The rest of the article is organized as follows: A discussion on portrayal of OS feature through UI in four major operating systems has been presented in Section 2 followed by a discussion on the issues with UI in Section 3. Section 4 demonstrates effectiveness of HMD as the futuristic UI. In particular, a mathematical analysis for improvement of FOV using HMD is presented in Section 4.1. In Section 5, we present some of the works that motivated us. Finally, we conclude the paper in Section 6.

## 2 OS Functionality through UI

As mentioned in the previous section, modern operating systems might need to portray essential functionalities of the system to the users through UI. In this section we explore one such essential functionality, namely, file system navigation. We have selected four popular (and major) OSs, among which two are for desktop systems and two are for mobile devices. These operating systems are Windows, Linux, Android, and iOS. In particular, we have selected Windows 10, Ubuntu 14.04 LTS, Android 6.0 Marshmallow, and iOS9. Following is a brief discussion on how directories, files, and related information are portrayed through UI by the OSs.

ISBN: 1-60132-433-2, CSREA Press ©

## 2.1 Windows 10

Windows has been the most preferred OS among naïve and some of the advanced users, but this success can arguably be attributed to successful marketing strategy rather than an intuitive user interface. The file system is navigated through Windows Explorer that hasn't change much over time. It is presented as a window with a scroll box of directories on the left. This box has categories such as "*Favorites*" which are infinitely customizable. Just right of the scroll box there is an area where icons are displayed and represent files or directories. Files might have different icons depending on the extension and directories are visible by a simple click.

As the name implies, in Windows, processes are portrayed within a window and the operation of this process is contained within a frame so that it is easy for users to differentiate between a process and the operating system. When developing a windows application, there are many tools available that help the developers portray the experience they desire. Tabs, charts, and drop-down-boxes are commonplace in Windows Forms Applications and they are relatively easy to understand. The execution and resources of each running process is displayed within the "*Task Manager*" where users can view the status of a process or can even kill it. Pressing "*SUPER + TAB*" can switch between processes and all running interactive applications are displayed in an assortment of windows. Alternatively, "*ALT + TAB*" can be pressed to display an icon-based list of virtual desktops. These features are useful even with a small screen as users can switch between different tasks quickly.

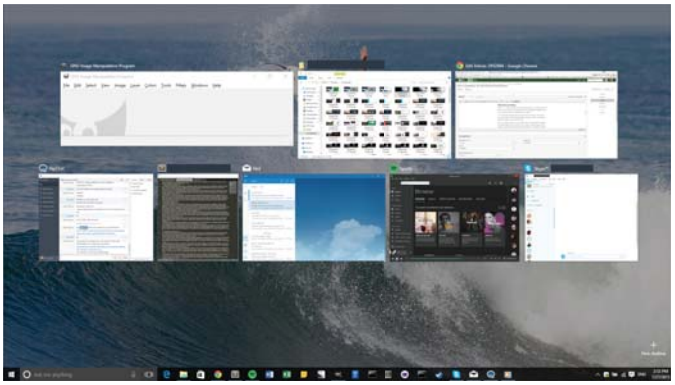


Figure 1. Example of Windows 10 Task View [7]

## 2.2 Ubuntu 14.04 LTS

Ubuntu 14.04 LTS comes with Unity interface that is surprisingly intuitive yet innovative. In many ways Unity is similar to the Windows desktop experience. There is a launcher bar that is persistently displayed showing all open applications and there are some icons to launch more applications. Windows are used to portray a running process that users can interact with. There is an application called "*System Monitor*" that allows users to view the resources being used by each process and they can kill processes from here. Ubuntu's application switching environment known as "*Workspaces*" is also very useful; it allows users to bring up different assortments of windows and save their states. Each assortment is switchable by a grid and by default there are multiple tiles as shown in Figure 2.

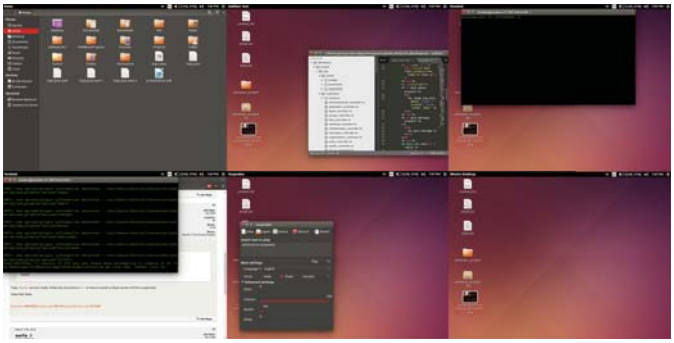


Figure 2. "Workspace" implemented in Ubuntu

## 2.3 Android 6.0 Marshmallow

Android 6.0 Marshmallow is the latest iteration of mobile operating systems from Google. File explorer designs for mobile operating systems need to be much different than desktop operating systems. The reason is twofold: (a) the devices with mobile operating systems have limited screen size, and (b) the use of touch-screen technology as a way to interact with the device. The file explorer in Android 6.0 is definitely not intended to be a commonly used feature. It is hidden in the *Options* menu and offers a simple interface where directories are listed as a vertical list (shown in Figure 3). Upon entering a directory, files are displayed as tiles with small previews of what each file contains. Interactive processes are not in "windows" but usually exist as non-scalable tiles that occupy the whole screen. It is possible to split the screen and display multiple applications at once, but this is highly limited by screen space. Application switching is extremely simple with this operating system. The swipe up and down action is bound to bring up a screen where running applications are displayed horizontally across the screen. The status of each application is displayed in small squares and has a visual representation of its UI included in the frame.

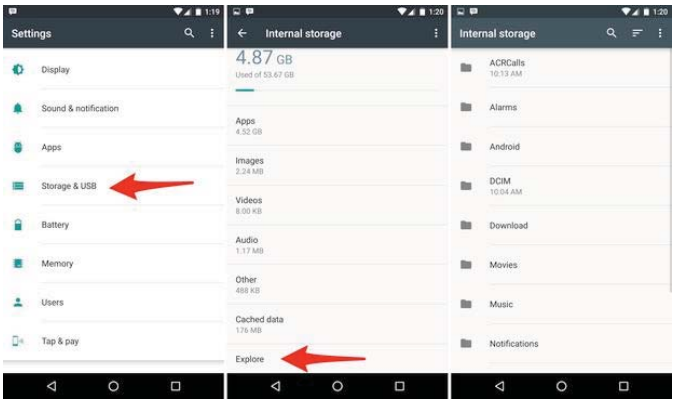



Figure 3. Android 6.0 Marshmallow file manager [8]

### 2.4 iOS9

iOS9 offers no intuitive way to browse the integral file structure of the system, rather Apple encourages cloud file hosting. Nonetheless, there are individual applications to browse type of structures. By not offering a way for the users to browse the file system of iOS9 device, Apple has simplified the user experience but sacrificed a key component of the UI of an operating system. Apple introduced a new way to switch applications on the release of iOS9, after double tapping the *home* button, open applications are displayed as three dimensional skewed frames with representative screens of the open UI. This presentation of switching is similar to the "WINDOWS + TAB" method introduced in Windows 7. There is no vanilla way to access the CPU usage or running processes of an iOS9 device. This is possibly to simplify the user experience but again limits the purpose of the operating system as a medium between the user and hardware.

### 2.5 Reflections

From the above discussion we conclude that modern operating systems rely heavily on the hardware (screen size) for providing functionalities to the user through UI. There are arguably many applications that can change the functionality of each of these operating systems, but as mentioned before, we are exploring the UI of each OS in a vanilla state.

## 3 Issues with UI of Modern OSs

In this section we intend to highlight the issue of limiting factors due to the UI of an operating system where the software and hardware is capable of performing a task, but the UI makes it difficult or impossible for a user to execute it. For example, as mentioned in Section 2.4, there is no easy way for a user to explore the file system of an iOS9 device. Perhaps this is to justify the "ease of use" benefit, but this implies that exploring a file system is difficult in the first place. This is an example of the sort of problems that can be solved with an innovative and easy to understand UI.

When there is a useful or innovative way to portray the UI of an operating system, it seems many companies are quick to release similar variations of the same design idea in order to maintain market presence. When the first iPhone was released along with iPhone OS 1.0, there was a shift in the market from traditional number pad controlled mobile operating systems to interactive touchscreen operating systems [1]. While this innovation might often be attributed to the software, the operating system of previous mobile devices was truly limited by the hardware. By making a large touch screen that engulfs the entire front of the device, Apple found boundless flexibility in what could be displayed and therefore could produce a simple and easy to use operating system that became immensely popular. The main point that should be deduced from this is that the introduction of a new hardware can enable huge innovations in software and thus solve some of the limiting factors of user interface of the past.

Amongst the operating systems examined in this paper, there are trends of limiting factors that are prevalent. File exploration is an integral function of an operating system, however, it has not changed much since the first window based user interfaces. There are indications that the modern implementations of this feature are difficult for new users to understand. For example, Android 6.0 has this feature hidden in the options menu and Apple has decided to completely omit it from iOS9. When we analyze the way file systems are portrayed to the user we see that it is done on a directory dependent basis where only the files and sub-directories contained within the current directory are displayed. This can be confusing to a new user because it is barely representative of the actual file structure of a system. More advanced users recognize that a file structure is like a tree where directories organize related and sub-related content. If the user interface of a file explorer more closely mimicked the actual structure of a file system, it would be easier for a user to understand the content being portrayed to them.

Part of an operating system is its ability to manage and run processes both for the user and itself. Many of the user processes are presented on a screen with a graphical user interface. This has been the trend especially for desktop operating systems and is a relatively intuitive way of interacting with an application. However, due to each application's share of screen, it is nearly impossible to manage more than a few applications at once on screen without having some way to switch between them. We have analyzed the way Windows 10 and Ubuntu 14.04 attempt to solve this problem by offering virtual desktops or arrays of open windows that are switchable. While it is true that our eyes might only be able to view a single window at once, the movement of our eyes to view another application is faster and much more intuitive than task switching on a single display. This is why many users choose to operate with multiple monitors on a desktop. While multiple monitors enhance the scope of UI in terms of coverage of field-of-view (FOV) of the user (provide better visibility), it is not space-effective, as the user needs to find place to set up multiple monitors on the his/her desktop. Therefore, just as Apple's innovation in mobile computing, solving this problem will require the introduction of a new hardware.

## 4 Future of User Interface with HMDs

The main purpose of an UI is to allow the users easier use of the computer system through abstraction. This abstraction started with text form (command line), evolved to GUI (including colors and icons), and now evolving to "hands-free" technologies like voice or gesture controls. Along this line, in the past few years head mounted displays (or HMDs) have caught the attention of many consumers and developers as a new way to view or display contents. HMDs like HoloLens [2], Google Glass [3], PlayStation VR [4], and Oculus Rift [5] use virtual augmentation of the surroundings through a headset. These HMDs have stereoscopic displays enabling each eye of the user to see a different image from the other, thereby enabling three-dimensional objects to be

represented to the user by a computer. Additionally, most HMDs offer tracking in both translation and orientation. This allows a computer to track the movement and position of a user's head when wearing the display.

Figure 4. Oculus Rift VR [9]

The operating systems and user interface supporting such devices can be categorized to provide "virtual reality" (hence VR). In this type of UIs, elements can be put anywhere within the 3D space and allows users to interact with UI elements in an intuitive way. Thus, users and designers get more freedom and can treat UI elements as physical objects. This offers an improvement on our current two-dimensional desk-mounted displays by adding the third axis (z-axis) using a high field-of-view depth sensor that tracks the user's 3D space. From this we can conclude that increasing the field-of-view would mitigate the challenges encountered by traditional 2D user interfaces.

In the next section, we calculate the amount of improvement an HMD could bring over a traditional two-dimensional monitor. In particular, we compare the FOV provided by both to measure the improvement.

## 4.1 Enhancing FOV using HMD

### 4.1.1 Field of View

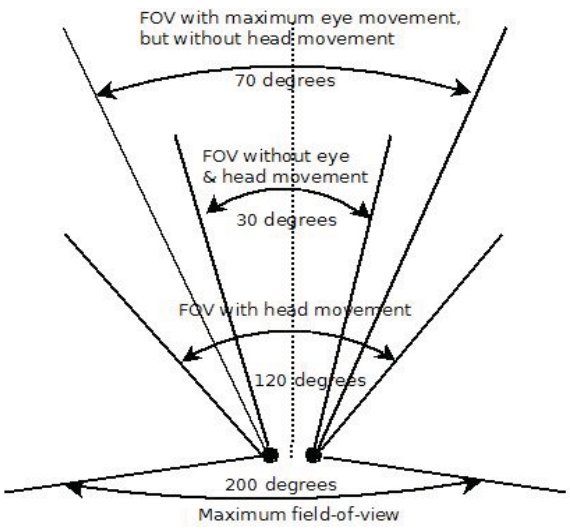



Figure 5. Field-of-View of human eye

Field-of-view is defined as the extent to which an optical sensor (like eyes) can see the observable environment. For human eyes the typical limit is 200° horizontally. This angular range includes both head and eye movement. Figure 5 shows the filed of view of human eyes in different conditions.

### 4.1.2 Enhancing FOV

Assuming the typical distance of a desktop monitor from a user sitting at a desk be 1 meter, we imagine an 1 meter circle drawn around a user's head. Calculating the distance around such a circle (circumference), say *cir*, is

$$cir = 2\pi \times 1 \text{ meter} = 2\pi \text{ meter} \quad (1)$$

Next, we calculate the representation of the possible field of view by the user. The average human field of view is around 200° horizontally [6]. To calculate amount of the circle viewable by the user, we divide the normal filed of view by 360°. Thereby we get the amount of circle visible $V_m$ as

$$V_m = 200° / 360° = 5/9 \quad (2)$$

Hence, the fraction of the circle viewable ($FOV_n$) by the user without movement of eye or head is calculated as

$$FOV_n = V_m \times cir \quad (3)$$

which is, $(5/9) \times 2\pi = 3.490658504... \approx 3.5$ meter

Equation 3 shows that the amount of 1 meter circle viewable around head of the user at a sitting position in front of a traditional two dimensional monitor. The purpose of this calculation is to estimate the amount of improvement by an HMD over a monitor. Both an HMD and traditional monitor do not use the entire field of view of human eyes and therefore the efficiency of each must be calculated against the complete field of view of the user.

We assume that an arbitrary monitor size, commonly shipped with modern desktops, is that of 27 inches or about 68 centimeters. The percentage of the user's FOV that is being utilized by a 68cm or 0.68 meter monitor placed a meter away from the user's eyes is given as

$$0.68/FOV_n = 0.19481948056503... \approx 20\% \quad (4)$$

Now we calculate the percentage of FOV utilized by an HMD. One modern HMD that offers impressive specifications is the *Oculus Rift* [5]. With the Rift, a user can experience up to 110 degrees field-of-view horizontally [6]. Following the same logic as used in establishing Equation 2, the correct result for the amount of circle visible inside the HMD, say $V_{HMD}$ is

$$V_{HMD} = 110° / 360° = 11/36 \quad (5)$$

Hence, the FOV for HMD is obtained as

$$FOV_{HMD} = (11/36) \times 2\pi = 1.92 \text{ meter} \quad (6)$$

Therefore, percentage of user's FOV that is being utilized by an HMD-based UI is

$$FOV_{HMD}/FOV_n = 1.92/3.5 = 0.54857142857 \approx 55\% \quad (7)$$

Comparing the result to that of the amount of FOV utilized by the monitor, in this particular case, the HMD is more than twice as efficient as the monitor. The advantage of an HMD is illustrated by mapping the FOV as seen in Figure 6.

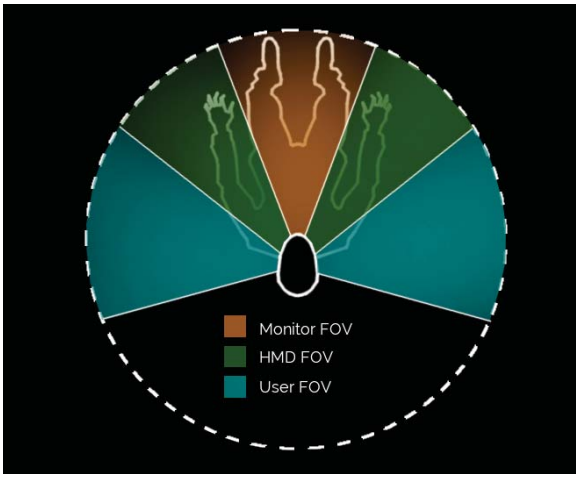



Figure 6. FOV of hardware vs. User

### 4.2 Adding third dimension using HMD

In addition to giving the user an increase in FOV, one must also consider that a head mounted display presents stereoscopic images. Most of the current user interfaces are two dimensional, however, when granted with a third axis user interfaces could be much more efficient at displaying information. In the earlier example of a file explorer, it was proposed that a more natural way of portraying file structure would benefit the users.

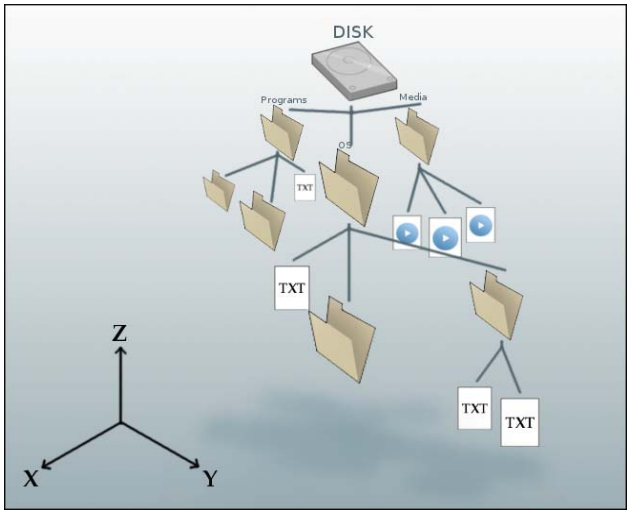



Figure 7. Suggested 3D depiction of file system

With an HMD, file structure can be presented as a three dimensional tree where directories and files are nodes in the tree. The alternate directories and structures are also visible but not the focal point of the structure. This 3D tree structure of the file systems is suggested in Figure 7.

## 5 Related Works

User interface design in an operating system is not a new area of research but the focus on innovative user interface design, especially three dimension UI design, is shifted relatively later. In one of the earlier papers, Hand [10] examined correspondence between improvement in the interaction techniques and the performance of 3D graphics hardware. [11] is another early work in literature that suggested a 3D UI design. The paper discussed effect of virtual environment hardware on user interactions. [12] is one of the earlier surveys that has been done on success and failures of traditional UI designs. It also suggested the need for 3D design of UI, customization capability by end users, and emphasized on speech or camera-based interaction systems. More recently interaction techniques in 3D environments have been reviewed in [13].

Effect of head mounted displays on filed of view has been investigated for a while. In [14] the author investigated performance of FOV with HMD on locating a target by turning head and moving through a maze-like environment. The experiments were conducted on wide FOV HMDs. Comparison of distance judgment in real world and virtual world viewed with HMD has been experimentally investigated in [15]. Enhancing horizontal FOV by using "catadioptrical HMD" and "omnidirectional image sensor" has been proposed in [16]. The FOV enhancement is achieved by placing ellipsoidal and hyperboloidal curved mirrors within the HMD. Not only the effect of HMD, the effect of cognitive load and age on field of view has also been investigated in the field of psychology [17].

Though these works provided new directions in UI designs, they did not focus on operating system functionalities, especially the file system navigation. The ideas presented in these works motivated us to start thinking about futuristic UI for operating systems that would incorporate some of the ideas and hardware mentioned above and would improve functionalities of an OS.

## 6 Conclusions

The importance of UI in an operating system cannot be overstated. User interfaces can solve or create problems depending on the implementation and the presentation of the workings of an operating system to a user. Increasing user's filed-of-view is one solution to current limitations. Introducing a third dimension into a user interface is another obvious place for improvement. Virtual reality devices like head mounted displays can provide both solutions: it enhances the user's field-of-view, and can provide a 3D interface to operating system features. The authors acknowledge that perhaps HMDs are not the solution to all UI

issues, but certainly can provide a direction to the advancement of I/O hardware to support significant improvement. Modern computing power would easily support a three dimensional operating system but it would be difficult to portray it on a two dimensional display. HMDs can mitigate this challenge. That way, these devices can find efficient usability outside gaming world and can provide UI support in more generic sense. It may also lead to a paradigm shift in computing.

# 7 References

[1] Lev Grossman. "Invention of the year: The iPhone". Time Magazine, Online 1, 2007.

[2] Microsoft HoloLens. Online, Accessed: April 30, 2016. https://www.microsoft.com/microsoft-hololens/en-us.

[3] Google Glass. https://www.google.com/glass/start/

[4] PlayStation VR: $399 coming in October, and it looks pretty good to us. http://www.cnet.com/products/sony-playstation-vr/

[5] Oculus Rift. https://www.oculus.com/en-us/

[6] Leonard J. Williams. "Cognitive load and the functional field of view"; Human Factors: The Journal of the Human Factors and Ergonomics Society, Volume 24, Issue 6, 683-692, 1982.

[7] Windows 10 taskview. http://www.pcworld.com/article/2952864/windows/how-to-use-windows-10s-task-view-and-virtual-desktops.html

[8] Jason Ciriani/CNET. "Marshmallow Screenshot". http://www.cnet.com/how-to/how-to-access-android-6-0-marshmallows-file-manager/

[9] David Ewalt. "Oculus Rift Review: The Beginning of the Age of VR". http://www.forbes.com/sites/davidewalt/2016/03/28/oculus-rift-review-the-beginning-of-the-age-of-vr/#1a29ddee1793

[10] Chris hand. "A Survey of 3D Interaction Techniques". Computer Graphics Forum, Volume 16, Issue 5, 269 – 281, December 1997.

[11] Doug Bowman, Ernst Kruijff, Joseph LaViola, and Ivan Poupyrev. "An Introduction to 3-D User Interface Design". MIT Press Journal, Volume 10, Issue 1. 96 – 108, February 2001.

[12] Brad Myers, Scott Hudson, and Randy Pausch. "Past, Present, and Future of User Interface Software Tools". ACM Transactions on Computer-Human Interaction – Special Issue on Juman-Computer Interaction in The New Millenium, Part 1. Volume 7, Issue 1, 3 – 28, March 2000.

[13] J. Jankowski and M. Hachet. "Advances in Interaction with 3D Environments". Computer Graphics Forum, Volume 34, Issue 1, 152 – 190, February 2015.

[14] Kevin Arthur. "Effects of Filed of View on Performance with Head-Mounted Displays". Ph.D. Thesis, Dept. of Computer Science, University of North Carolina at Chapel Hill, 2000.

[15] Peter Willemsen, Mark Colton, Sarah Creem-Regehr, and William Thompson. "The Effects of Head-Mounted Display Mechanical Properties and Field of View on Distance Judgments in Virtual Environments". ACM Transactions on Applied Perception. Volume 6, Issue 2 Article no. 8, February, 2009.

[16] Hajime Nagahara, Yasushi Yagi, and Masahiko Yachida. "Wide Filed of View Head Mounted Display for Tele-presence with An Omnidirectional Image Sensor". In Computer Vision and Pattern Recognition Workshop, Madison, USA. Volume 7, 86 – 91, June 2003.

[17] Richard Park. "An Investigation of Perceptual Load, Aging, and The Functional Filed of View." Ph.D. Thesis, School of Psychology, Georgia Institute of Technology, 2005.