UNIVERSITY OF CALIFORNIA,
IRVINE

Ensuring Reliability and Fault-Tolerance for the Cyber-Physical System Design

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Volkan Gunes

Dissertation Committee:
Professor Tony Givargis, Chair
Professor Alexandru Nicolau
Associate Professor Ian G. Harris
Dr. Steffen Peter

2015

DEDICATION

I would like to dedicate this dissertation to my family for their invaluable support and unconditional love throughout my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help of so many people in so many ways. It is the product of my educational, professional, and personal attainments through my discussions with people.

Among those with whom I have had very fruitful discussions, I would like to express my deepest appreciation to my final defense committee chair and advisor, Prof. Tony Givargis, who provided me with the guidance and support in all aspects of my research at the University of California, Irvine. I learned a lot from his guidance and invaluable suggestions.

I would also like to thank Dr. Steffen Peter for his great suggestions related to my research, which enabled me to explore possible research directions. The insightful discussions with him opened several prospects to me and enriched my research contributions.

I would like to thank my candidacy and final defense committee members, Prof. Alexandru Nicolau, Prof. Ian G. Harris, Prof. Nikil Dutt, and Prof. Ahmed M. Eltawil for their time and insightful comments.

I would like to thank my research group members and other colleagues at the University of California, Irvine. The useful discussions with them always helped me to find my way in my research. I also would like to thank ICS student affairs who helped me navigate the complexities of formal procedure in graduate school.

I would like to thank my friends for many interesting discussions and enjoyable experiences, and for helpful insights on research, graduate school, and daily life matters. I thank all of you for all the good times that I have had with you, and all your support and best wishes in the difficult times.

I would like to thank IEEE, the copyright holder of some of the materials included in Chapter 1, Chapter 3, Chapter 4 and Chapter 5, and KSII, the copyright holder of the material included in Chapter 2, for granting me the permission to use my own publications as part of this dissertation.

I gratefully acknowledge the National Science Foundation for their support of the research contained in this dissertation via NSF grant numbers 1016789 and 1136146. I would like to express my sincere appreciation to the Ministry of National Education in Turkey for their support during my education in the United States.

Finally, and most importantly, I would like to express my deepest gratitude to my parents, who gave me encouragement and supported me with their best wishes throughout my study at the University of California, Irvine. Their guidance and support have always helped me to overcome the challenges that I have faced throughout my life. Therefore, I cannot thank them enough for all they have done for me.

CURRICULUM VITAE

Volkan Gunes

EDUCATION

Doctor of Philosophy in Computer Science **2015**
University of California, Irvine *Irvine, California, USA*
Donald Bren School of Information and Computer Sciences

Master of Science in Computer Science **2010**
University of California, Irvine *Irvine, California, USA*
Donald Bren School of Information and Computer Sciences

Bachelor of Science in Electronics **2006**
Gazi University *Ankara, TURKEY*
Electronics and Computer Education Department

RESEARCH EXPERIENCE

Graduate Student Researcher **2012–2015**
University of California, Irvine *Irvine, California, USA*
The Center for Embedded and Cyber-physical Systems (CECS)

TEACHING EXPERIENCE

Teaching Assistant **2013–2014**
University of California, Irvine *Irvine, California, USA*
Donald Bren School of Information and Computer Sciences

PUBLICATIONS

V. Gunes, S. Peter, and T. Givargis, “Modeling and Mitigation of Faults in Cyber-Physical Systems with Binary Sensors”, in Proc. of the 16th IEEE International Conference on Computational Science and Engineering (CSE), pp. 515-522, December 2013.

V.Gunes and T.Givargis, “XGRID: A Scalable Many-Core Embedded Processor”, in the Center for Embedded Cyber-physical Systems (CECS) at UCI, Technical Report #TR 13-03, 2013.

V. Gunes, S. Peter, T. Givargis, Frank Vahid, “A Survey on Concepts, Applications, and Challenges in Cyber-Physical Systems”, in KSII Transactions on Internet and Information Systems, volume 8, issue 12, pp. 4242-4268, December 31, 2014.

V. Gunes and T. Givargis, “XGRID: A Scalable Many-Core Embedded Processor”, in Proc. of the 12th IEEE International Conference on Embedded Software and Systems (ICESS), August 2015.

V. Gunes, S. Peter, T. Givargis, “Improving Energy Efficiency and Thermal Comfort of Smart Buildings with HVAC Systems in the Presence of Sensor Faults”, in Proc. of the 12th IEEE International Conference on Embedded Software and Systems (ICESS), August 2015.

PEER REVIEW ACTIVITIES

The Int’l Conf. on Compilers, Arch., and Synthesis for Embedded Sys. (CASES)	**2011**
The Int’l Conference on Computer-Aided Design (ICCAD)	**2011**
The Int’l Conf. on Compilers, Arch., and Synthesis for Embedded Sys. (CASES)	**2012**
The Int’l Conf. on Hardware - Software Co-design and Sys. Synthesis (CODES-ISSS)	**2012**
The 10th IEEE Symp. on Embedded Systems for Real-time Multimedia (ESTIMedia)	**2012**
The 18th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS)	**2012**
The 50th ACM Design Automation Conference	**2013**
The 51th ACM Design Automation Conference	**2014**
The Int’l Conf. on Hardware - Software Co-design and Sys. Synthesis (CODES-ISSS)	**2014**
The 13th IEEE Symp. on Embedded Systems for Real-time Multimedia (ESTIMedia)	**2015**

ABSTRACT OF THE DISSERTATION

Ensuring Reliability and Fault-Tolerance for the Cyber-Physical System Design

By

Volkan Gunes

Doctor of Philosophy in Computer Science

University of California, Irvine, 2015

Professor Tony Givargis, Chair

The cyber-physical system (CPS) is a term describing a broad range of complex, multi-disciplinary, physically-aware next generation engineered systems that integrate embedded computing technologies into the physical world. Sensors play an important role in this integration because they provide the data extracted from the physical world for the cyber systems. However, this process is likely to be misled by incorrect data due to sensor faults.

In this dissertation, the main focus is on sensor fault mitigation and achieving high reliability in CPS operations. One of the challenges we tackle is timely event detection in CPS under faulty sensor conditions. In this regard, we examine the falling ball example (FBE) using binary event detectors, a controller, and a camera for timely motion detection and estimation of a falling ball. Another challenge we tackle is satisfying thermal comfort and energy efficiency under faulty sensor conditions in a multi-room building incorporating temperature sensors, controllers, and heating, ventilation, and air conditioning (HVAC) systems. For both cases, we adopt a model-based design (MBD) methodology to analyze the effect of sensor faults on the system outcome. In this regard, we develop well-defined fault and system evaluation models and incorporate them into the traditional CPS model that comprises the cyber, interface (e.g., sensors and actuators) and physical models.

We explore various fault mitigation techniques based on redundancies and temporal-spatial correlations between sensors' data in a holistic design perspective. Furthermore, considering compute demands of CPSs, we introduce the XGRID embedded many-core architecture. XGRID makes use of a novel, FPGA-like, programmable interconnect infrastructure offering scalability and deterministic communication. We further introduce a deployment scenario of XGRID as a use case for thermal control of the multi-room building.

Our findings regarding reliable CPS design show that the physical system attributes can be more dominant than the cyber system attributes on the system outcome. In addition, sensor faults may lead to unsatisfactory system outcome since CPSs heavily rely on sensor readings for decision making. Therefore, the analysis of temporal and spatial correlations between sensor readings helps mitigate sensor faults and enable CPSs to utilize sensors' data more efficiently for decision making.

Chapter 1

Introduction

1.1 The Motivation

Embedded computer systems touch nearly every aspect of our daily lives. Demand for embedded computer systems have increased in recent years for applications including but not limited to consumer electronics (e.g., smart TVs, smart phones, etc.), industrial automation (e.g., process control systems), transportation systems (e.g., autonomous automotive systems and air vehicles), military applications (e.g., unmanned air vehicles, weapon control systems, etc.), and medical systems (e.g., insulin pump, medical ventilator, etc.). This increasing demand for embedded computer systems brings forth an important requirement for their deployments in the real world, that is the physical awareness. Embedded computer systems have to be physically-aware in order to help us get the utmost benefits from our real life experiences.

The next generation engineered systems, namely what have become known as the cyber-physical systems (CPSs), tightly couple sensing, networking, and embedded computing technologies in order to provide awareness and control of the physical world. As an important constituent of CPSs, the state-of-the-art sensor technologies allow extracting helpful knowledge from the physical world. Since the physical systems constitute an important portion of CPSs, CPSs heavily rely on the knowledge extracted by various sensors from those physical systems and their environment. Any sensor fault that occurs during the readings from the physical systems impairs the knowledge extracted from the physical systems and is most likely to compromise CPS reliability unless appropriate measures are taken. In this dissertation, the main focus is on sensor fault mitigation and achieving high reliability in CPS operations. The approaches presented in this study contribute to the design of fault-tolerant CPSs.

1.2 The Problem Definition

The design of a CPS requires a significant amount of reasoning with respect to unique challenges, and complex performance and reliability requirements. Hence, dependability and reliability are two important requirements for CPS design. However, the potential component failures may prevent CPSs from performing dependable and reliable operations. Such failures may occur in the cyber part, the physical part, or the interface part (i.e., sensors, actuators, etc.) and must be analyzed at the early stages of design.

Failure is an inevitable incident for applications in the physical world. Failure semantics specifies the behavior of a system in case a failure occurs. To ensure the satisfaction of a system outcome, potential failures must be handled to guarantee the expected behavior of the system. First of all, such failures need to be identified and categorized. Later, the solutions for them should be addressed in a systematic way. This study primarily focuses on the sensor interface of CPSs and addresses the following questions:

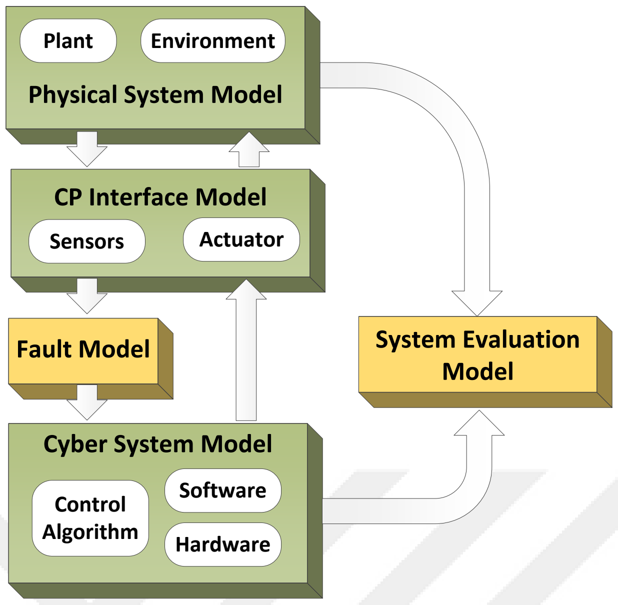


Figure 1.1: Model-based CPS design methodology.

- Which faults can occur in the sensor interface of CPSs between the cyber and the physical parts?
- How can we cope with those faults?

The first question leads to the task of understanding fault semantics and modeling possible faults, while the second question concerns fault mitigation strategies.

1.3 The Methodology

In this study, we adopt a model-based approach to design reliable CPSs. Our model-based CPS design methodology is shown in Figure 1.1. A common CPS model consists of the physical system model, the cyber-physical (CP) interface model, and the cyber system model, as represented with green rectangles in Figure 1.1. The physical system model can be expressed as the description of physical behavior with differential equations. Those equations should reflect both inner dynamics (i.e., plant behavior and conditions) or outer dynamics (i.e., conditions of environment surrounding the plant) of the physical system. The cyber-physical (CP) interface model represents the network of sensors and actuators. The cyber system model can be expressed as control algorithm, and the behavior of software and hardware components of the cyber system.

Since this study researches CPS performance evaluation under certain types of sensor faults and sensor fault mitigation, we clearly define fault semantics and system evaluation metrics, develop models for sensor faults and system evaluation, and inject those models into the traditional CPS model. By doing so, we aim to examine the behavior of a CPS under imperfect sensor conditions and explore the design space by considering trade-offs in design decision. Sensor placement and temporal-spatial correlations between sensor readings are throughly examined and highlighted as an important factor in CPS design. This study addresses reliable and fault-tolerant CPS design with the emphasis on sensor fault specification and mitigation.

1.4 The Contributions

In this study, we examine the challenge of tolerating sensor faults and designing reliable CPS under imperfect sensor conditions. The primary contributions of this study can be summarized as follows:

- **CPS concepts and challenges:** We explore various CPS concepts including CPS retrospective, terminology, and similar research topics. We bring existing research efforts and initiatives regarding CPS design into the readers' attention, including efforts in various application domains. We provide the taxonomy of CPS challenges and our findings regarding reliable CPS design.

- **Model-based design methodology:** We propose a model-based design methodology for CPS applications and develop reusable system models in the MATLAB/Simulink environment. Adopting model-based design approach facilitates integrated system development. Further, model-based development creates an environment conducive to easy integration of fault and system evaluation models into the traditional CPS model for sensor fault analysis.

- **Delivering time-critical operations in CPS:** We propose fault mitigation techniques for timely event detection, specify fault semantics for event detectors (e.g., motion detectors), explore timing uncertainties, address the effects of sensor placement on the reliable system operation. Component and algorithm redundancy approaches are introduced considering spatial correlations between sensors' data. We introduce a design space exploration approach based on those faults and other important design factors (e.g., cost) for fault mitigation.

- **Design of a multi-room building with CPS approach:** We introduce the design of a multi-room building with CPS approach. A number of sensor data faults observed in the real-world sensor deployments are examined. The read-back and nearest neighbor monitoring approaches are proposed considering temporal and spatial correlations between sensors' data. We develop fault models in the MATLAB/Simulink environment, which can be leveraged across other CPS applications.

- **A many-core embedded processor for CPS compute demands:** We propose a many-core embedded processor, namely what we call as XGRID, to fulfill compute demands of large scale CPSs. We introduce a novel, FPGA-like, programmable interconnect infrastructure, offering scalability and deterministic communication using hardware supported message passing among cores. In addition, we propose an application mapping algorithm based on Kahn process networks (KPNs) and integer linear programming (ILP) to aid in the mapping of applications on XGRID. Further, we provide a conceptual mapping of control algorithms for the automation of a multi-room building onto target XGRID architecture as a use case of XGRID in CPS.

The ultimate goal of this study is to design a complete, trustworthy, reliable, and fault-tolerant CPSs that are capable of achieving high standards expected from them.

1.5 The Organization

This study consists of six chapters including this chapter. The rest of the study is organized as follows:

Chapter 2 provides a comprehensive point of view on CPSs by covering history and definitions; terminology and concepts related to CPS; domains, applications, and existing efforts in each domain; system-level challenges and requirements to realize CPS vision. The content of Chapter 2 was published in [1].

Chapter 3 deals with timely event detection in a CPS under certain fault occurrences. In this chapter, we explain sensor fault semantics and our model-based design approach for a use case application to examine the impact of such faults on the system outcome. We discuss fault mitigation techniques for those faults, and design space exploration under certain criteria (i.e., cost and fault types). A portion of Chapter 3 was published in [2].

Chapter 4 deals with the thermal model of a multi-room building from CPS perspective. Temperature sensor fault semantics from real-world data sets are introduced and implemented in MATLAB/Simulink using model-based design approach. Our methodology to tolerate those specified sensor faults are explained by considering energy efficiency and thermal comfort of the occupants as evaluation metrics for the impact of faults on the system outcome. A portion of Chapter 4 was published in [3].

Chapter 5 deals with XGRID many-core embedded processor for CPS compute demands. We introduce XGRID architecture and describe an application mapping algorithm based on Kahn process networks (KPNs) and integer linear programming (ILP) to aid in the mapping of applications on XGRID. We further introduce a deployment scenario of XGRID as a use case for thermal control of a multi-room building. A portion of Chapter 5 was published in [4].

Chapter 6 concludes the study by providing insights derived from the research conducted regarding trustworthy, reliable, and fault-tolerant CPS design.

Chapter 2

The Cyber-Physical System (CPS)

2.1 Introduction

Advances in digital electronics have led to a significant increase in the number of systems that couple the digital (cyber) systems with the physical world, namely what have become known as the cyber-physical systems (CPSs). This chapter presents the history and various definitions of CPS; terminology and concepts related to CPS; applications, domains, and existing efforts in each domain; system-level challenges and requirements to realize CPS vision. The aim of this chapter is to provide sufficient information regarding CPSs in the form of a survey, which helps the reader gain a holistic and comprehensive view of CPSs and understand what considerations needed to be taken for trustworthy CPS design.

The design of CPSs requires a significant amount of reasoning with respect to unique challenges and complex functional, reliability, and performance requirements. A number of articles have addressed the necessary problem formulations, system-level requirements, and arising challenges in CPS design. Baheti and Gill [5] introduce CPS concept and suggest research directions for CPS design. Lee [6] specifically points out the failure of standard abstraction layers, the need of reliable timing behavior, and lack of temporal semantics of existing programming language models for CPS design. Rajkumar [7] touches on system level aspects of CPS from the scientific and social impact standpoints. Lee [8] suggests two approaches, namely cyberizing the physical and physicalizing the cyber, for integrating the cyber systems with the physical systems.

A variety of existing surveys describe the holistic view of CPSs. Shi [9] gives an outline of CPS features, challenges, and applications without going into the details. Sanislav and Miclea [10] describe CPS specifications, design, and research directions and briefly cover CPS applications and system level requirements. Horvath and Gerritsen [11] touch on CPS characteristics, design technologies (i.e., cyber, physical, and synergic technologies), and implementation principles.

The incentive for us to conduct this survey arises from a lack of unifying concepts, definitions, related terminologies, challenges, and applications. We aim to provide sufficient insight into CPS concepts and common applications. In this chapter, we discuss the followings:

- CPS history, applications and challenges.
- Concepts similar to CPS.
- A glimpse of CPS application domains and existing efforts in each domain to realize CPS vision.

The rest of this chapter is organized as follows. CPS history and definitions are presented in Section 2.2. CPS terminology and concepts relatively similar to CPS are explained in Section 2.3. Domains and applications of each domain are introduced in Section 2.4. CPS challenges are discussed in Section 2.5. Conclusions are provided in Section 2.6. The content of this chapter was published in [1].

2.2 CPS History and Definitions

CPS is an emerging area that refers to the next generation engineered systems. The term CPS was coined at the National Science Foundation (NSF) in the United States around 2006 [12]. The CPS approach has been recognized as a paramount and prospective shift towards future networking and information technology (NIT) by the 2007 report of the President's Council of Advisors on Science and Technology (PCAST). PCAST recommends the reorganization of the national priorities in NIT research and development (R&D) and putting CPSs at the top of the research agenda [13]. The National Science Foundation (NSF) has increasingly provided funding opportunities to the scientific community to promote transformative research on CPSs [14].

A special interest organization has been set up in the U.S., namely Cyber-Physical Systems Virtual Organization (CPS-VO), to foster collaboration among CPS professionals in academia, government, and industry [15]. The European Union's joint technology initiative, called Advanced Research and Technology for Embedded Intelligence Systems (ARTEMIS), has invested in research and development (R&D) efforts on the next generation engineered systems with public-private partnership between European Nations and the industry to fulfill the vision of a world in which all systems, machines, and objects become smart and physically-aware, have a presence in the cyber-physical space, exploit the digital information and services around them, and communicate with each other as well as with the environment [16]. Moreover, the European Commission has launched a new research and innovation program, namely Horizon 2020, at the end of 2013 to develop new strategies for tackling societal challenges. Horizon 2020 is the biggest research and innovative program yet with a budget of nearly EUR 80 billion. Horizon2020 covers CPSs and advanced computing research and innovation [17].

CPS has been defined by the scientific community from different perspectives. Rajkumar [7] describes CPSs as “physical and engineered systems, whose operations are monitored, coordinated, controlled, and integrated by a computing and communicating core”. Lee [18] describes CPSs as “integrations of computation with physical processes”. Marwedel [19] describes them as “embedded systems together with their physical environment”. Gill [20] describes them as “physical, biological, and engineered systems whose operations are integrated, monitored, and/or controlled by a computational core. Components are networked at every scale. Computing is deeply embedded into every physical component, possibly even into materials. The computational core is an embedded system, usually demands real-time response, and is most often distributed”.

In summary, the cyber-physical systems (CPSs) are complex, multi-disciplinary, physically-aware next generation engineered systems that integrate embedded computing technology (cyber part) into the physical phenomena by using transformative research approaches. This integration mainly includes observation, communication, and control aspects of the physical systems from the multi-disciplinary perspective.

2.3 CPS Terminology and Relatively Similar Concepts

Although CPS is a relatively new concept, the system components are well-known. As shown in Figure 2.1, CPS is composed of the physical world, the interfaces, and the cyber systems. The physical world refers to the physical phenomena wanted to be monitored or controlled. The cyber systems refer to the next generation embedded devices, which process information and communicate with their distributed environment. The interfaces refer to the communication network and other intermediate components, e.g., interconnected sensors, actuators, analog-to-digital converters (ADC), and digital-to-analog converters (DAC), responsible for bridging the cyber systems with the physical world. Sensors and actuators are responsible for

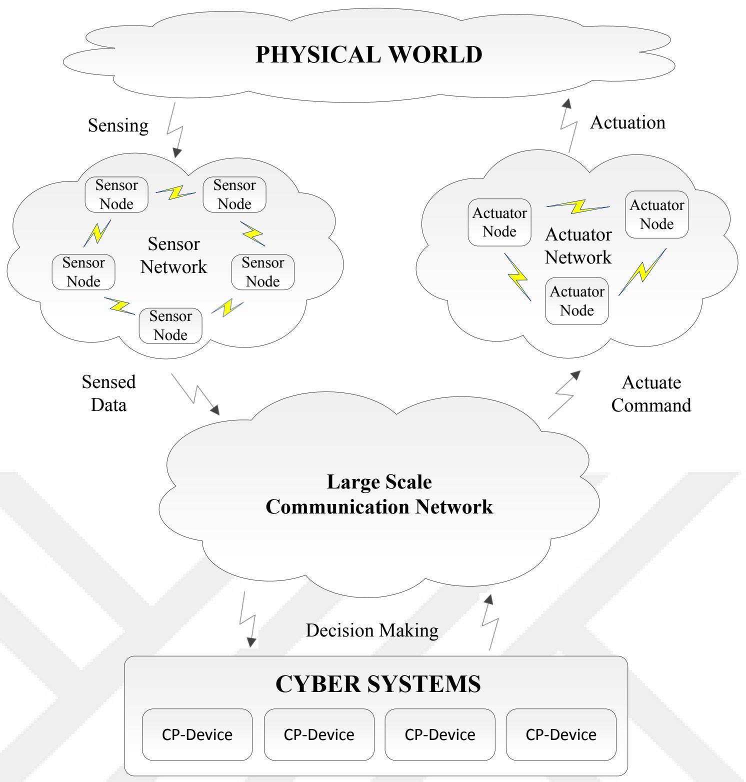


Figure 2.1: CPS holistic view.

converting other forms of energy to electricity (analog signal) and vice versa, respectively. ADC and DAC are responsible for converting continues analog signals to discrete digital signals and vice versa, respectively.

Resource scheduling in shared sensor and actuator networks (SANs) is a challenging task and plays an important role in CPS operation. In this regard, actuation coordination is essential to decide which actuators must be scheduled to perform a particular action or how to manage control actions properly. Various parameters, such as actuator capabilities, real-time guarantee, task completion time, energy consumption of each actuator, and the physical system requirements must be considered during control task allocation to particular actuator [21].

Regarding actuator scheduling, an important difference of CPSs compared to most cyber systems is the reversibility or preemption of actuator operations. While in most cyber systems, roll-back operations and preemption is available (e.g., databases or bus access protocols), physical operations executed by actuators typically cannot be reversed. If an actuation is performed based on erroneous data, it is often very challenging or impossible to roll back the activity, as for instance discussed in [22] for specific health care applications. Additionally, non-reversibility challenge affects real-time scheduling in cases where several jobs are managed on a shared platform. Even hard real-time tasks may be blocked by low-priority processes if a shared actuation resource access cannot be preempted or rolled back, as for instance discussed for a satellite communication system [23].

The control aspect of the physical phenomena and the theory behind control systems is a basis for all state-of-the-art continuous time dynamical systems and thus has a crucial role in CPS design. Conventionally, control policies are completely separate from the system infrastructure and implemented after manufacturing the system prototype [24]. Such an approach is not feasible to meet the demands expected from CPSs because of their complex and dynamic nature. To meet those demands and perform complex control laws, the physical system itself and its dependency relationship with those control laws should be well defined and modeled [25].

CPSs must operate in real-time. Real-time control is traditionally implemented through different forms of control mechanisms, namely open-loop control, feed-forward control, and feed-back control. The open-loop control strategy utilizes only the input signal (desired value) to actuate the output according to the control requirements and lacks a feed-back mechanism to adjust the output of the system, therefore expects adjustments manually from the operator [26]. The feed-forward control strategy considers environmental effects measured via sensors over the physical system. Then, the control action is adjusted by the controller according to the anticipation of the relationship between the physical system and its environment [26]. The feed-back (a.k.a., closed-loop) control strategy automatically refines the output based on the difference between the feed-back signal from the output and the input signal. Both the physical system and the controller affect each other, hence the name closed-loop. All environmental effects (e.g., disturbance) on the physical system are taken into account via the feed-back signal [26]. Since CPS applications incorporate the physical systems/environments, and interact with them through their physical-awareness capability without human intervention, many of them are likely to adopt the feed-forward and the feed-back strategies together at the lowest level.

In case the feed-back loop is closed over wireless sensor and actuator networks (WSANs), passivity-based control design can be applied to make the control design insensitive to network uncertainties (e.g., time-varying delays) [27]. Fidelity-aware utilization control, which integrates data fusion with the feed-back control, can be adopted in wireless cyber-physical surveillance systems to optimize system fidelity and adaptively adjust the control objective of CPU utilization in the presence of environmental variations (e.g., noise characteristics) [28]. The importance of control theory in CPS design has been addressed by a number of studies [5, 7, 12, 29, 30, 31, 32, 33, 34].

Conventionally, if the feed-back control of a system is closed through a shared network, then that system is called a networked control system (NCS), in which the control input/plant output is passed through interconnected system components (such as sensors, controller, and actuators) [35]. Another type of control systems is called SCADA, which stands for supervisory control and data acquisition. These types of control systems are utilized to monitor and control processes, including but not limited to industrial, infrastructure, and facility-based processes that exist in the physical world. A SCADA system gathers data in real time from sensors in local and remote locations and transfers them to the central computers in order to control the equipment/conditions and take necessary actions [36]. CPSs entail requirements far beyond the expectation of legacy control systems, such as NCS and SCADA.

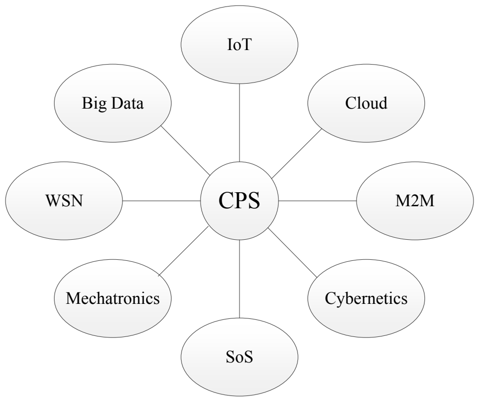


Figure 2.2: Similar concepts.

Some core concepts in CPS can be traced back to the sensor network research and technologies related to sensor nodes and sensor networks. A sensor node integrates sensors, actuators, computing elements (e.g., processor, memory, etc.), communication modules, and a battery. The sensor network interconnects many small sensor nodes via wireless or wired connection [37]. Called as wireless sensor networks (WSNs), a large number of sensor nodes equipped with wireless network connection can be deployed in the environment of the physical phenomenon. Those sensor nodes may provide raw data to the nodes responsible for data fusion or they may process the raw data by means of their computing capabilities and relay the required part of it to the other sensor nodes.

Various research areas and terminologies are relatively similar to CPS. A range of concepts similar to CPS is illustrated in Figure 2.2. The term big data refers to the datasets that are too large and complex to capture, store, manage, and analyze with standard methods or database tools [38]. A large scale CPS can be envisioned as millions of networked smart devices, sensors, and actuators being embedded in the physical world, which can sense, process, and communicate the data all over the network. Proliferation of technology-mediated social interactions via these highly featured and networked smart devices has allowed many individuals to contribute to the size of big data available. Depending on the size of data sets and number of smart devices involved, big data may be in the range of multiple terabytes to many petabytes (i.e., 1024 terabytes) [39].

The term cloud is a paradigm shift in information and communications technology (ICT), through which businesses and users can have an on-demand network access to a shared pool of configurable computing resources (e.g., hardware, applications, services, etc.) [40, 41]. Cloud computing model promotes broad network access to a pool of resources, optimal usage and control of resources, minimal management effort of hardware and software resources, scalable computing capabilities, and on-demand services without human interaction with service providers [40]. Cloud computing provides new opportunities for CPSs in management and processing of aggregated sensor data and decision making methods based on a cloud model allow CPSs to enhance the system capability.

The term system of systems (SoS) refers to large-scale, heterogeneous systems networked together for a common goal and composed of inherently autonomous components that can be operated and managed independently [42]. The term has been addressed by the systems engineering community and reflects the interest in large-scale systems that have considerable economical and societal impacts (e.g., critical infrastructures, intelligent transportation, emergency response, etc.) [43].

The term mechatronics is the combination of mecha, referring to mechanical systems, and tronics, referring to electronic systems. The term was coined in the late 1960s. However, it has evolved over the decades comprising software and information technologies. Therefore, it can be considered as a systematic approach to design, develop, and implement complex engineering systems which incorporate information technologies into the physical domain [44, 45].

The term cybernetics refers to an approach describing the study of communication and control characteristics in both machines and in living beings [46]. Broadly speaking, cybernetics involves the qualitative analysis of the relationship between various system components and whole system behavior [47]. The theory of cybernetics and the practice of mechatronic system design lay the foundations for the design of CPSs [48].

Inspired by the idea of interconnecting smart devices, the term Internet of things (IoT) was coined in 1999. IoT was envisioned as a future radio frequency identification (RFID) technology that enables the automatic identification of the physical objects via a small electronic chip called RFID tag. IoT provides an opportunity to observe, identify, and understand the real world by capturing data about the things (i.e., RFID tagged objects) and help businesses achieve greater efficiency and accountability [49, 50].

IoT greatly overlaps with CPS, because IoT addresses observing the things in the physical world, exploiting communication capabilities, and capturing data needed to manage the things that are not efficiently managed today [49, 51]. Even though IoT originally targeted identification and monitoring technologies, today IoT also applies to the control of the physical systems by the integration of RFID systems and sensor networks, namely RFID sensor networks [52].

The several important aspects of IoT are surveyed in detail in [53]. The survey includes different perspectives of IoT, revision of enabling technologies with the emphasis on what is being done and what needs to be done for further research. Besides IoT, the idea of interconnecting several heterogeneous CPSs under a large-scale universal network (like the Internet) is addressed in [54] and referred to as the cyber-physical Internet (CPI).

Inspired by IoT, the web of things (WoT) integrates real world objects (things) into the World Wide Web using standard web technologies. In WoT, each physical object that contains an embedded device is identified as a standard web resource with URI and can be accessed through web APIs [55]. This provides connectivity of embedded devices at the application layer. A five-layer WoT framework to integrate WoT and CPS is studied in [55], using an intelligent vehicle system as a case study.

Machine-to-machine (M2M) communication is another concept related to CPS. M2M refers to smart devices, such as computers, embedded processors, smart sensors, actuators, and mobile devices, talking to each other via a communication network [56, 57]. M2M is a communication standard that is a subset of both IoT and CPS. Existing research on M2M communications are surveyed in [58] from architecture, standard development, and representative application perspective. The authors also propose a solution to the integration of intelligent road and unmanned vehicle with wireless sensor networks (WSNs) navigation in the form of CPS. Enabling new business models, both M2M and IoT target data aggregation by offering smart services to customers to improve efficiency and provide automation and low-cost systems in the world of e-commerce [50, 57].

The above-mentioned concepts clarify why the National Intelligence Council (NIC) foresees IoT/CPS as one of the six disruptive civil technologies with potential impacts on the U.S. interests [59]. The next generation Internet technologies are expected to play an important role on both IoT and CPS research. Therefore, how we interact with the real world will probably be revolutionized just like the traditional Internet revolutionized how we interact with one another [7, 60].

2.4 Domains and Applications

Various studies have addressed the domains and domain specific applications of CPSs. In this section, we summarize a number of research efforts that address some of those domains, namely smart manufacturing, emergency response, air transportation, critical infrastructure, health care and medicine, intelligent transportation, and robotic for service. With this

Table 2.1: Functionality of CPS domains.

Type of Domain	**Scale/Functionality**
Smart manufacturing	Medium scale; optimizing productivity in the manufacture of goods or delivery of services.
Emergency response	Medium/large scale; handling the threats against public safety, and protecting nature and valuable infrastructures.
Air transportation	Large scale; operation and traffic management of aircraft systems.
Critical infrastructure	Large scale; distribution of daily life supplies such as water, electricity, gas, oil.
Health care and medicine	Medium scale; monitoring health conditions of the patients and taking necessary actions.
Intelligent transportation	Medium/large scale; improving safety, coordination and services in traffic management with real-time info sharing.
Robotic for service	Small/medium scale; performing services for the welfare of humans.
Building Automation	Medium scale; optimizing control and automation of HVAC, lighting, fire prevention, security, and entertainment systems in the buildings.

summary, we aim to cite a few of the recent research efforts from CPS perspective for each application domain. Table 2.1 provides an overview on the CPS applications according to their functionality. More details on the CPS applications are given in the following subsections.

2.4.1 Smart Manufacturing (SM)

Smart manufacturing refers to the use of embedded software and hardware technologies to optimize productivity in the manufacture of goods or delivery of services [61]. Smart factory is another frequently mentioned concept to refer to the next generation smart manufacturing.

Smart manufacturing is one of the leading CPS application domains because of drivers like mass production, domestic and international marketing, economic growth, etc. A large effort on characterizing CPS for smart manufacturing has been undertaken in Europe and the U.S. The Industrie 4.0 project is a German strategic initiative, which represents a major opportunity for manufacturing of the future [62]. The Industrie 4.0 is aimed to take a pioneering role in manufacturing of the future. A non-profit organization, namely the Smart Manufacturing Leadership Coalition (SMLC), was established in the U.S. SMLC involves manufacturing supplier, practitioner, and consortia, technology companies, universities, and government labs that have expressed interest in realizing smart manufacturing of the future [63].

Over the years, manufacturing confronts with lots of demands for high flexibility. It is very challenging to meet those demands today because of safety reasons. Those safety reasons arise from close interactions and co-operations between machines and human experts in the absence of sufficient sensors and intelligent devices to avoid possible accidents [64]. CPS perspective on the future industrial revolution will improve safety, productivity, and efficiency by connecting embedded system production technologies to pave the way to highly flexible work flow and new forms of collaboration [65].

2.4.2 Emergency Response (ER)

Emergency response refers to handling the threats against public safety, health, and welfare and protecting the nature, properties, and valuable infrastructures. CPSs can provide fast emergency response via large number of sensor nodes in the regions in case of the natural or man-made disasters. However, this rapid response requires the nodes to collectively assess the situation and rapidly inform the central authority even in the frequently-changing environments. So robustness, effective resource utilization, adaptiveness, and timeliness come into play in this emergency response [66].

Emergency response and disaster management have always drawn attention because of their societal implications. A White House Presidential Innovation Fellow project, namely the SmartAmerica Challenge, was launched in December of 2013 in the U.S [67]. The project is aimed to bring industry, academia, and the government together in the CPS agenda and to gather research efforts, projects, and activities from different domains together. Disaster response is one of the domains/challenges in the SmartAmerica Challenge.

The Strategic Foresight Initiative (SFI) was launched by the Federal Emergency Management Agency (FEMA) in the U.S. Department of Homeland Security (DHS). According to preliminary research results conducted by SFI, aging infrastructures pose a potential risk for the emergency management because they become less reliable and hinder disaster recovery [68].

Emergency management of the future should adopt emerging information technologies and social media use. Strong collaboration and cooperation among emergency management professionals, local and national authorities, and the community are needed. The use of effective forewarning, response, and recovery mechanisms are required in the future emergency management systems [69]. Unmanned aerial or ground vehicles can be deployed to provide efficient search and rescue efforts. Besides, new embedded technologies having physical awareness need to be integrated into the infrastructures to manage emergency response and disaster recovery in the future.

2.4.3 Air Transportation (AT)

Air transportation refers to any civil or military aviation systems and their traffic management. Smart air vehicles are expected to be predominant in the near future, especially for military service. The unmanned aerial vehicle (UAV), commonly known as the drone, is just one of the well-known examples of smart air vehicles. Since physical-awareness is an important issue for the next generation air vehicles, CPSs are expected to make a profound impact on the future aviation and air traffic management (ATM) [5, 31].

Distributed control throughout the airspace is expected to become a substantial part of the next generation ATM systems. However, that would give rise to more scalability challenges since interactions between vehicles and infrastructure are becoming more complicated. Current capacity constraints at the major airports and airspace interactions between the airports and air vehicles in a multi-airport system limit the overall capacity of the system [31, 70].

The operation of aircrafts has been regulated over years by the procedures similar to those specified over 30 years ago [71]. Today, air traffic control is managed through radar towers and computing support systems have limited physical awareness. So, tight integration of the computational and physical capabilities is of paramount importance for the next generation air transportation systems.

As an existing effort to realize air transportation of the future, a vision of the next generation air transportation, namely NextGen, is introduced by the Federal Aviation Administration (FAA) in the U.S. Department of Transportation. NextGen is an approach transforming air traffic control from routing over radar towers to routing over satellite-based technology [72]. Satellite navigation is used to provide the pilots with precise locations of surrounding airplanes. New system is being installed step by step and it is expected to see the outcomes of the approach (e.g., flight costs, enhanced safety, etc.) by the year 2018 [72].

2.4.4 Critical Infrastructure (CI)

Critical infrastructure refers to valuable properties and public infrastructures that are necessary for the survival or welfare of the nations. The smart grid is one of the appealing applications in the critical infrastructure domain. The smart grid incorporates central/industrial power plants, energy storage and transmission facilities, renewable energy resources (such as wind farms and solar cells), and energy distribution and management facilities in smart homes/buildings [73].

The smart grid describes the transformation from a centralized, producer-controlled network of electricity grid to a less centralized, more distributed, more cooperative, more responsive, and more consumer-interactive one by bringing future information and communication technologies and power system engineering together for grid modernization [74].

The smart grid provides real-time load monitoring, distribution, and planning at utility level; a balance of supply and demand at the device level; two-way flow of information (i.e., real-time communication between the consumer and utility); the integration of existing energy resources into the grid; large scale grid awareness and ability to switch between high level (e.g., state-wide) and low level (e.g., street-wide) grid exploration; real time integration of sensor data with geographical information; and power quality and blackouts monitoring as well as prevention or minimization of a potential outage [75].

Besides the smart grid, water distribution is another important service for the communities. The SmartAmerica Challenge project introduces an enhanced water distribution infrastructure challenge enabled by cellular based CPS that will eventually provide real-time monitoring of water quality and flow control; faster response to possible contamination; low cost and more secure water; and better leak detection [67].

2.4.5 Health Care and Medicine (HC&M)

Health care and medicine refers to the issues addressing multiple aspects of the patient's physiology. A special attention is drawn to medical applications in CPS research since they provide significant research opportunities for the CPS community. These opportunities include, but are not limited to, technologies related to home care, assisted living, smart operating room, smart medical devices (e.g., pace maker, medical ventilator, infusion pump, etc.), and smart prescription [5, 32].

Current technological trends and challenges in the design of the cyber-physical medical system (CPMS) are summarized in [76] along with promising research directions. These trends cover reliable software-based development to deliver new functionalities, increased connectivity of medical devices equipped with network interfaces, and demand for continuous patient monitoring (e.g., home care, assisted living, telemedicine, and sport-activity monitoring, etc.). Modeling and model-driven engineering will play an important role in the future CPMS development [76].

Interoperability as a system level requirement is of paramount importance in CPMS development. There have been joint efforts to satisfy that requirement, for example the Medical Device Plug-and-Play (MD PnP) Interoperability program [77] which was established in 2004. This program leads open platform and standard developments for medical device interoperability. These developments will allow heterogeneous systems to be composed in plug-and-play fashion, increase patient safety, and enable new treatment options and proliferation of technology [78].

Today, medical technology only provides limited access and integration of data along with manual coordination of medical devices and loops are not closed [79]. The cyber-physical medical systems of the future should provide extensive data integration and access, comprehensive data acquisition and analysis, closed loop control capabilities, energy efficiency, real-time visualization, and plug-and-play capability with interoperable medical devices.

2.4.6 Intelligent Transportation (IT)

Intelligent transportation refers to the advanced technologies of sensing, communication, computation, and control mechanisms in transportation systems to improve safety, coordination, and services in traffic management with real-time information sharing. Intelligent transportation facilitates both ground and sea transportation through information sharing over satellites and provides communication environment among vehicles, the infrastructure, and passengers' portable devices [80].

The intelligent transportation systems (ITSs) integrate pedestrians, vehicles, sensors, roadside infrastructures, traffic management centers, satellites, and other transportation system components by adopting different variation of wireless communication technologies and standards [81]. ITSs of the future allow real-time traffic monitoring; increase in transportation safety and comfort through information exchange among traffic users; optimal traffic management; collision avoidance; and utilization of satellite based technology to connect drivers, roads, and vehicles smoothly [82].

With the integration of CPSs into infrastructures, vehicles, and roadways, ITSs can achieve driver assistance, collision avoidance or notification, improvements in travel time without fear of unexpected delays, reductions in congestion, and advanced control over infrastructure and vehicles for energy saving [71]. ITSs rely not only on advanced sensor and embedded computer systems technology but also on wireless, cellular, and satellite technologies for vehicle-to-vehicle (V2V), vehicle-to-pedestrian (V2P), and vehicle-to-infrastructure (V2I) communication to better manage complex traffic flow, ensure safety, and extend situational awareness.

2.4.7 Robotic for Service (RfS)

Robotic for service refers to deploying intelligent robots to perform services for the welfare of humans, and the equipment in a fully autonomous, semi-autonomous, or remotely controlled manner, excluding manufacturing operations [83]. Robotic for service is identified as one of the six disruptive civil technologies with potential impacts on the U.S. interests out to 2025 [59].

Robots can be deployed for several purposes, including but not limited to defense (e.g., explosive disposal, surveillance in prohibited areas, etc.), environment monitoring and control, assisted living, logistics, and so on. Since the next generation robots are likely to have close interactions with humans in the physical environment of their operation, learning and interpretation of human activities by the robots comes into play as an important factor. From CPS perspective, integration of humans and smart robots is very important to enable all actors of CPSs to achieve better cooperation, collaboration, and organization to overcome complex duties [84].

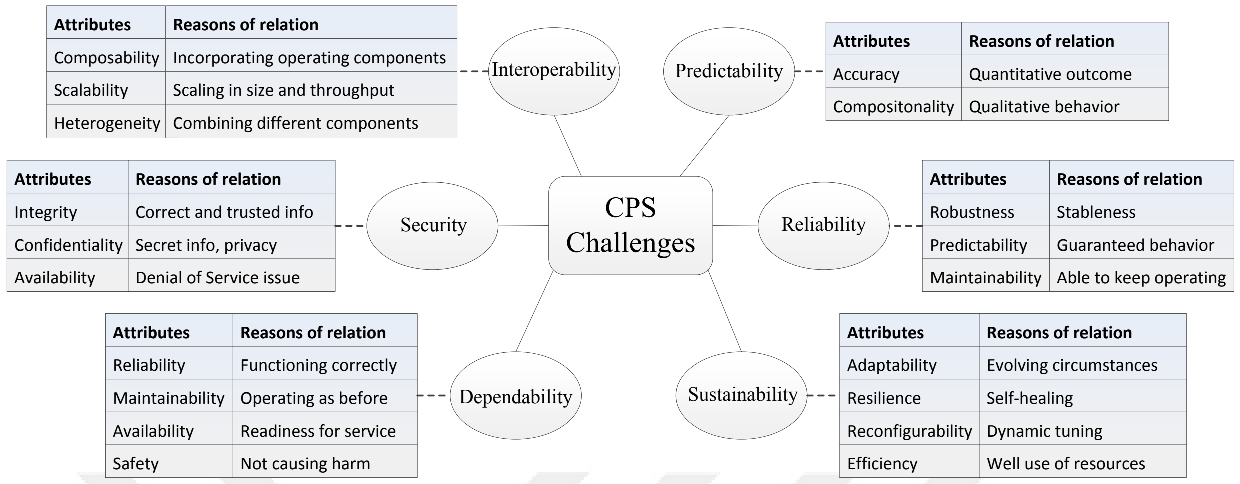
To realize the vision of the next generation robotics, the National Science Foundation (NSF), in partnership with the National Institutes of Health (NIH), the U.S. Department of Agriculture (USDA), and the National Aeronautics and Space Administration (NASA) launched a new initiative, namely the National Robotics Initiative, and makes new investments totaling approximately $38 million in the development of the next generation robots to achieve cooperation and collaboration between humans and robots for enhanced productivity [85]. Another initiative, namely the Robotics Virtual Organization (Robotics-VO), has been launched in 2012 in the U.S. The initiative provides information for the members of the robotics community about funding opportunities, conferences, principal investigator (PI) meetings, robotics news and seminars, and educational resources [86].

2.4.8 Building Automation (BA)

Building automation refers to the deployment of various sensors, actuators, and distributed control systems to provide optimum control and automation of heating, ventilation, and air conditioning (HVAC), lighting, fire prevention, and security systems in the buildings. Smart/intelligent building is a frequently mentioned concept to address the next generation buildings.

Smart buildings are needed to fulfill the vision of the smart grid and smart city concepts. With the growing popularity, IoT/CPS provides great opportunities for new applications in the next generation building automation concept via a large range of smart building appliances including entertainment media as well, which in return brings diverse requirements and interaction patterns for realizing such systems [87]. Besides being applied in homes and offices, building automation from CPS perspective can be applied to laboratories. Since activities done in laboratories have been getting sophisticated due to technological advances, new arrangements and services, such as regulation of environmental conditions due to environment-sensitive equipment, accessing incidents or abnormalities, tracing dangerous materials, harvesting energy etc., are needed for the management of laboratories in the future [88].

In the future, smart buildings are expected to pick up the slack for fulfilling the needs of connecting the smart grid with smart living environments and learning their living patterns to ensure comfortable living environments. Smart house case models developed by a number of universities in the U.S. as prototypes are reviewed in [89]. Findings are that the applications of smart buildings are closely connected with the deployment of intelligent technologies as of such day. However, smart buildings must be designed as being adaptable and responsive not only to the short term but also to the long term needs of the users considering technological and social changes [89].



Attributes	Reasons of relation
Composability	Incorporating operating components
Scalability	Scaling in size and throughput
Heterogeneity	Combining different components

Attributes	Reasons of relation
Accuracy	Quantitative outcome
Compositonality	Qualitative behavior

Attributes	Reasons of relation
Integrity	Correct and trusted info
Confidentiality	Secret info, privacy
Availability	Denial of Service issue

Attributes	Reasons of relation
Robustness	Stableness
Predictability	Guaranteed behavior
Maintainability	Able to keep operating

Attributes	Reasons of relation
Reliability	Functioning correctly
Maintainability	Operating as before
Availability	Readiness for service
Safety	Not causing harm

Attributes	Reasons of relation
Adaptability	Evolving circumstances
Resilience	Self-healing
Reconfigurability	Dynamic tuning
Efficiency	Well use of resources

Figure 2.3: CPS challenges.

Smart utility networks, such as home area networks (HANs), neighborhood area networks (NANs), wide area networks (WANs), and so on, can be deployed in the smart power distribution network of the future to provide two way flows of electricity and information. HAN communication can be utilized for building automation to deliver data traffic and control instructions not only between the smart utilities (e.g., smart meters) and the residents' smart devices but also between the residents' smart devices themselves [90].

2.5 CPS Challenges

Cyber-physical systems revolutionize our interaction with the physical world. Of course, this revolution does not come free. Since even legacy embedded systems require higher standards than general-purpose computing, we need to pay special attention to this next generation physically-aware engineered system requirements if we really want to put our full trust in them. Therefore, we want to clarify the definitions of some common CPS system-level requirements/challenges. We associate main CPS challenges with their attributes in Figure 2.3. Brief definitions for the challenges are given in the following and an overview linking challenges to applications concludes this section.

Dependability refers to the property of a system to perform required functionalities during its operation without significant degradation in its performance and outcome. Dependability reflects the degree of trust put in the whole system. A highly dependable system should operate properly without intrusion, deliver requested services as specified and not fail during its operation. The words dependability and trustworthiness are often used interchangeably [91]. Assuring dependability before actual system operation is a very difficult task to achieve. For example, timing uncertainties regarding sensor readings and prompt actuation may degrade dependability and lead to unanticipated consequences. Cyber and physical components of the system are inherently interdependent and those underlying components might be dynamically interconnected during system operation, which, in return, renders dependability analysis very difficult. A common language to express dependability related information across constituent systems/underlying components should be introduced in the design stage [92, 93].

Maintainability refers to the property of a system to be repaired in case a failure occurs. A highly maintainable system should be repaired in a simple and rapid manner at the minimum expenses of supporting resources, and free from causing additional faults during the maintenance process. With the close interaction among the system components (e.g., sensors, actuators, cyber components, and physical components) underlying CPS infrastructure, autonomous predictive/corrective diagnostic mechanisms can be proposed. Continuous monitoring and testing of the infrastructure can be performed through those mechanisms. The outcome of monitoring and testing facilities helps finding which units need to be repaired. Some components, which happen to be the source of recurrent failures, can be redesigned or discarded and replaced with the ones with better quality [94, 65].

Availability refers to the property of a system to be ready for access even when faults occur. A highly available system should isolate malfunctioning portion from itself and continue to operate without it. Malicious cyber-attacks (e.g., denial of service attacks) hinder availability of the system services significantly. For example, in the cyber-physical medical systems, medical data shed light on necessary actions to be taken in a timely manner to save a patient's life. Malicious attacks or system/component failure may cause services providing such data to become unavailable, hence, posing risk on the patient's life [95].

Safety refers to the property of a system to not cause any harm, hazard or risk inside or outside of it during its operation. A very safe system should comply with both general and application-specific safety regulations to a great extent and deploy safety assurance mechanisms in case something went wrong. For example, among the goals for smart manufacturing (SM), point-in-time tracking of sustainable production and real-time management of processes throughout the factory yield to improved safety. Safety of manufacturing plants can be highly optimized through automated process control using embedded control systems and data collection frameworks (including sensors) across the manufacturing enterprise. Smart networked sensors could detect operational failures/anomalies and help prevention of catastrophic incidents due to those failures/anomalies [63].

Reliability refers to the degree of correctness which a system provides to perform its function. The certification of system capabilities about how to do things correctly does not mean that they are done correctly. So a highly reliable system makes sure that it does the things right. Considering the fact that CPSs are expected to operate reliably in open, evolving, and uncertain environments, uncertainty in the knowledge, attribute (e.g., timing), or outcome of a process in the CPS infrastructure makes it necessary to quantify uncertainties during the CPS design stage. That uncertainty analysis will yield to effective CPS reliability characterization. Besides, accuracy of physical and cyber components, potential errors in design/control flow, cross-domain network connections in an ad-hoc manner limit the CPS reliability [96, 82].

Robustness refers to the ability of a system to keep its stable configuration and withstand any failures. A highly robust system should continue to operate in the presence of any failures without fundamental changes to its original configuration and prevent those failures from hindering or stopping its operation. In addition to failures, the presence of disturbances possibly arising from sensor noises, actuator inaccuracies, faulty communication channels, potential hardware errors or software bugs may degrade overall robustness of CPSs. Lack of modeling integrated system dynamics (e.g., actual ambient conditions in which CPSs operate), evolved operational environment, or unforeseen events are other particular non-negligible factors, which might be unavoidable in the run-time, hence the need for robust CPS design [97].

Predictability refers to the degree of foreseeing of a system's state/behavior/functionality either qualitatively or quantitatively. A highly predicable system should guarantee the specified outcome of the system's behavior/functionality to a great extent every moment of time at which it is operating while meeting all system requirements. In the cyber-physical medical systems (CPMSs), smart medical devices together with sophisticated control technologies are supposed to be well adapted to the patient's conditions, predict the patient's movements, and change their characteristics based on context awareness within the surrounding environment [33]. Many medical devices perform operations in real-time, satisfying different timing constraints and showing diverse sensitivity to timing uncertainties (e.g., delays, jitters, etc.). However, not all components of CPMSs are time-predictable. Therefore, in addition to new programming and networking abstractions, new policies of resource allocation and scheduling should be developed to ensure predictable end-to-end timing constraints [34].

Accuracy refers to the degree of closeness of a system's measured/observed outcome to its actual/calculated one. A highly accurate system should converge to the actual outcome as close as possible. High accuracy especially comes into play for CPS applications where even small imprecisions are likely to cause system failures. For example, a motion-based object tracking system under the presence of imperfect sensor conditions may take untimely control action based on incorrect object position estimation, which in return leads to the system failure [2].

Compositionality refers to the property of how well a system can be understood entirely by examining every part of it. A highly compositional system should provide great insight about the whole from derived behaviors of its constituent parts/components. Achieving high compositionality in CPS design is very challenging especially due to the chaotic behavior of constituent physical subsystems. Designing highly compositional CPSs involves strong reasoning about the behavior of all constituent cyber and physical subsystems/components and devising cyber-physical methodologies for assembling CPSs from individual cyber and physical components, while requiring precise property taxonomies, formal metrics and standard test benches for their evaluation, and well-defined mathematical models of the overall system and its constituents [96].

Sustainability means being capable of enduring without compromising requirements of the system, while renewing the system's resources and using them efficiently. A highly sustainable system is a long lasting system which has self-healing and dynamic tuning capabilities under evolving circumstances. Sustainability from energy perspective is an important part of energy provision and management policies. For example, the smart grid facilitates energy distribution, management, and customization from the perspective of customers or service providers by incorporating green sources of energy extracted from the physical environment. However, intermittent energy supply and unknown/ill-defined load characterization hinders the efforts to maintain long-term operation of the smart grid. To maintain sustainability, the smart grid requires planning and operation under uncertainties, use of real-time performance measurements, dynamic optimization techniques for energy usage, environment-aware duty cycling of computing units, and devising self-contained energy distribution facilities (such as autonomous micro grids) [98, 73].

Adaptability refers to the capability of a system to change its state to survive by adjusting its own configuration in response to different circumstances in the environment. A highly adaptable system should be quickly adaptable to evolving needs/circumstances. Adaptability is one of the key features in the next generation air transportation systems (e.g., NextGen). NextGen's capabilities enhance airspace performance with its computerized air transportation network which enables air vehicles immediately to accommodate themselves to evolving operational environment such as weather conditions, air vehicle routing and other pertinent flight trajectory patterns over satellites, air traffic congestion, and issues related to security [99].

Resilience refers to the ability of a system to persevere in its operation and delivery of services in an acceptable quality in case the system is exposed to any inner or outer difficulties (e.g., sudden defect, malfunctioning components, rising workload, etc.) that do not exceed its endurance limit. A highly resilient system should be self-healing and comprise early detection and fast recovery mechanisms against failures to continue to meet the demands for services. High resilience comes into play in delivering mission-critical services (e.g., automated brake control in vehicular CPSs, air and oxygen flow control over an automated medical ventilator, etc.). Mission-critical CPS applications are often required to operate even in case of disruptions at any level of the system (e.g., hardware, software, network connections, or the underlying infrastructure). Therefore, designing highly resilient CPSs requires thorough understanding of potential failures and disruptions, the resilience properties of the pertinent application, and system evolution due to the dynamically changing nature of the operational environment [92].

Reconfigurability refers to the property of a system to change its configurations in case of failure or upon inner or outer requests. A highly reconfigurable system should be self-configurable, meaning able to fine-tune itself dynamically and coordinate the operation of its components at finer granularities. CPSs can be regarded as autonomously reconfigurable engineered systems. Remote monitoring and control mechanisms might be necessity in some CPS application scenarios such as international border monitoring, wildfire emergency management, gas pipeline monitoring etc. Operational needs (e.g., security threat level updates, regular code updates, efficient energy management, etc.) may change for such scenarios, which calls for significant reconfiguration of sensor/actuator nodes being deployed or the entire network to provide the best possible service and use of resources [100].

Efficiency refers to the amount of resources (such as energy, cost, time, etc.) the system requires to deliver specified functionalities. A highly efficient system should operate properly under optimum amount of system resources. Efficiency is especially important for energy management in CPS applications. For example, smart buildings can detect the absence of occupants and turn off HVAC units to save energy. Further, they can provide automated pre-heating or pre-cooling services based on the occupancy prediction techniques [101]. Since various sensors are deployed in smart buildings to extract data from physical phenomena such as air temperature and quality, sensor faults may significantly reduce HVAC energy efficiency in smart buildings unless mitigated [3].

Security refers to the property of a system to control access to the system resources and protect sensitive information from unauthorized disclosures. A highly secure system should provide protection mechanisms against unauthorized modification of information and unauthorized withholding of resources, and must be free from disclosure of sensitive information to a great extent. CPSs are vulnerable to failures and attacks on both the physical and cyber sides, due to their scalability, complexity, and dynamic nature. Malicious attacks (e.g., eavesdropping, man-in-the-middle, denial-of-service, injecting fake sensor measurements or actuation requests etc.) can be directed to the cyber infrastructure (e.g., data management layer, communication infrastructure, decision making mechanisms, etc.) or the physical components with the intent of disrupting the system in operation or stealing sensitive information. Making use of a large-scale network (such as the Internet), adopting insecure communication protocols, heavy use of legacy systems or rapid adoption of commercial off-the-shelf (COTS) technologies are other factors which make CPSs easily exposed to the security threats [102, 103].

Integrity refers to the property of a system to protect itself or information within it from unauthorized manipulation or modification to preserve correctness of the information. A high integrity system should provide extensive authorization and consistency check mechanisms. High integrity is one of the important properties of a CPS. CPSs need to be developed with greater assurance by providing integrity check mechanisms on several occasions (such as data integrity of network packets, distinguishing malicious behaviors from the ambient noise, identifying false data injection and compromised sensor/actuator components etc.). Properties of the physical and cyber processes should be well-understood and thus can be utilized to define required integrity assurance [102, 104].

Confidentiality refers to the property of allowing only the authorized parties to access sensitive information generated within the system. A highly confidential system should employ the most secure methods of protection from unauthorized access, disclosure, or tampering. Data confidentiality is an important issue that needs to be satisfied in most CPS applications. For example, in an emergency management sensor network, attacks targeting confidentiality of data transmitted may degrade effectiveness of an emergency management system. Confidentiality of data transmitted through attacked sensor nodes can be compromised and that can cause data flow in the network to be directed over compromised sensors; critical data to be eavesdropped; or fake node identities to be generated in the network. Further, false/malicious data can be injected into the network over those fake nodes. Therefore, confidentiality of data circulation needs to be retained in a reasonable degree [105].

Interoperability refers to the ability of the systems/components to work together, exchange information and use this information to provide specified services. A highly interoperable system should provide or accept services conducive to effective communication and interoperation among system components. Performing far-reaching battlefield operations and having more interconnected and potentially joint-service combat systems, unmanned air vehicles (UAVs) call for seamless communication between each other and numerous ground vehicles in operation. The lack of interoperability standards often causes reduction in the effectiveness of complicated and critical missions [106]. Likewise, according to changing needs, dynamic standards should be developed and tested for devices, systems, and processes used in the smart grid to ensure and certify the interoperability of those ones being considered for a specific smart grid deployment under realistic operating conditions [107].

Composibility refers to the property of several components to be merged within a system and their inter-relationships. A highly composable system should allow recombination of the system components repeatedly to satisfy specific system requirements. Composibility should be examined in different levels (e.g., device composibility, code composibility, service composibility, system composibility). Certainly, system composibility is more challenging, hence the need for well-defined composition methodologies that follow composition properties from the bottom up. Additionally, requirements and evaluations must be composable accordingly. In the future, it will probably be of paramount importance to incrementally add emerging systems to the system of systems (e.g., CPS) with some predictable confidence without degrading the operation of the resulting system [108].

Heterogeneity refers to the property of a system to incorporate a set of different types of interacting and interconnected components forming a complex whole. CPSs are inherently heterogeneous due to constituent physical dynamics, computational elements, control logic, and deployment of diverse communication technologies. Therefore, CPSs necessitate heterogeneous composition of all system components. For example, incorporating heterogeneous computing and communication capabilities, future medical devices are likely to be interconnected in increasingly complex open systems with a plug-and-play fashion, which makes a heterogeneous control network and closed loop control of interconnected devices crucial.

Configuration of such devices may be highly dynamic depending on patient-specific medical considerations. Enabled by the science and emerging technologies, medical systems of the future are expected to provide situation-aware component autonomy, cooperative coordination, real-time guarantee, and heterogeneous personalized configurations far more capable and complex than today's [34].

Scalability refers to the ability of a system to keep functioning well even in case of change in its size/increased workload, and take full advantage of it. The increase in the system throughput should be proportional to the increase in the system resources. A highly scalable system should provide scatter and gather mechanisms for workload balancing and effective communication protocols to improve the performance. Depending on their scale, CPSs may comprise over thousands of embedded computers, sensors, and actuators that must work together effectively. Scalable embedded many-core architectures with a programmable interconnect network can be deployed to deliver increasing compute demand in CPSs [4]. Further, a high performance and highly scalable infrastructure is needed to allow the entities of CPSs to join and leave the existing network dynamically. In the presence of frequent data dissemination among those entities, dynamic software updates (i.e., changing the computer program in run-time) can help update CPS applications dynamically and use CPS resources more productively [109].

In fact, all system requirements/attributes are related to each other although we point out the general requirements and their direct/immediate attributes. We aimed to highlight the holistic view of system requirements for CPS applications in Figure 2.3. CPS is a very broad research area and therefore has diverse applications spanning different scales. All of CPS applications need to be designed considering its impact on the real world and necessary system-level requirements.

Table 2.2: References' table that touches on the importance of the subject challenges for CPS application domains.

Domain	**Dependability**	**Sustainability**	**Security**	**Reliability**	**Interoperability**	**Predictability**
AT	[103, 110]	[98, 111]	[98, 110, 112, 113]	[106, 111, 114]	[70, 106]	[33, 113]
CI	[33, 103]	[73, 98]	[33, 73, 98, 112, 115, 116, 117, 118]	[33, 73, 119]	[33, 73, 107]	[73]
ER	[103]	[120]	[98, 105, 120, 121]	[121, 122, 123]	[121, 123]	[122]
SM	[33, 103]	[124]	[33]	[33, 125]	[33, 125]	[125]
HC&M	[34, 103, 126]	[33, 98]	[66, 98, 112]	[34]	[33, 34, 127]	[34]
IT	[33, 128]	[33]	[115, 128]	[129]	[33]	[33]
RfS	[103]	[130, 131]	[131]	[132]	[130]	[133]

Table 2.2 addresses existing studies that touch upon the importance of those requirements for the application domains. Even though it cannot reflect the overall CPS research, it gives us a notion that the trend of CPS research is on critical infrastructures, specifically the smart grid, and security aspects of them.

2.6 Conclusions

The cyber-physical system (CPS) is a promising paradigm for the design of current and future engineered systems and is expected to make an important impact on our interactions with the real world. The idea behind CPS places the focus on the integrated system design instead of on the cyber or the physical system independently. In order to shed some light on the origins, the terminology, relatively similar concepts, and today's challenges in CPS, we presented this survey on related literature discussing practical applications and dominant research domains. Since CPS is a very broad research area, CPSs span diverse applications in different scales. Therefore, each application necessitates strong reasoning capabilities with respect to unique system-level requirements/challenges, the integration of cutting-edge technologies into the related application, and overall impact on the real world. We conclude that existing legacy systems have limited awareness of the CPS requirements, and that revolutionary design approaches are necessary to achieve the overall system objectives.

Chapter 3

Sensor Fault Modeling and Mitigation for Event Detection in CPS

3.1 Introduction

In recent years, advances in embedded system and sensor technologies have sparked a considerable interest in tight integration of the physical world with the digital world. This interest led to the term cyber-physical systems (CPSs) that, as the name implies, consist of the physical and cyber systems, coordinated together. In this tight integration of the physical world with the digital world, the sensors play an important role for reliable CPS operations. CPS applications deploy a wide range of sensors as an interface between the physical world and the cyber systems. Event detectors are among those sensors. Examples of event detectors include motion detectors, break beam sensors, pressure mats, etc. Motion detectors are used in CPS applications to identify location, velocity, or trajectory of a physical target.

Reliably delivering time-critical services is an important requirement of many CPSs. This requires profound understanding of the attributes of the cyber and physical system components. To ensure the success of time-critical functions for event detection in CPSs, potential sensor faults and timing uncertainties that lead to unreliable operations must be identified during the design phase of CPSs. In this chapter, we examine reliable motion detection in a CPS application. Our demonstrative example of a CPS is the falling ball example (FBE) [134], using event detectors (i.e., motion detectors). In this chapter, we follow the model-based CPS design methodology introduced in Chapter 1. We specify sensor fault semantics based on probabilities and examine the impact of the sensor faults on the system outcome. The analysis of system success is conducted to assess the overall performance of the system by introducing a system evaluation metric and model.

The rest of this chapter is organized as follows. Motivation and related work are presented in Section 3.2. Terminology and fault semantics are introduced in Section 3.3. A demonstrative example of a CPS is introduced in Section 3.4. Model-based design of the demonstrative example is introduced in Section 3.5. Fault mitigation techniques are introduced in Section 3.6. Experiments and evaluation are discussed in Section 3.7. Finally, Section 3.8 concludes the chapter. A portion of this chapter was published in [2].

3.2 Related Work and Motivation

Various authors have studied event detection mechanisms under fault occurrences and timing uncertainties. An application of smart transportation is studied in [135]. The focus of the study is to enhance time-critical intelligent services in the smart car. The authors identify the events which require timely responses and aim to provide services to guarantee system's timeliness.

An intruder tracking system is proposed in [136] to automatically track and capture photos of an intruder via a camera. In this study, the authors consider the problem of automated position estimation using a wireless network of inexpensive binary motion sensors to track an intruder. A high level robot behavior is evaluated in [137] under the presence of erroneous sensors. The authors do probability based analysis to see if the controllers satisfy a set of high level specifications for robotic behaviors.

A trustworthy alarm system, namely Tru-Alarm, is proposed in [138] to detect intrusion objects in a bounded environment. Tru-Alarm is responsible for filtering meaningful information from a large volume of data including erroneous ones caused by sensor failures. A target tracking problem with wireless sensor networks is studied in [139]. The authors propose a distributed tracking algorithm that estimates target's location, velocity and trajectory under the imperfect sensor conditions.

A real-time user tracking system, namely FindingHuMo (Finding Human Motion), is proposed in [140]. The system can accommodate tracking of multiple (unknown and variable number of) users in any crowded Smart Environments. The system can detect and isolate motion trajectories of individual users via anonymous (not user- specific) binary motion data stream.

Besides the above-mentioned applications, event detectors can be deployed in smart home/ buildings. For example, a garage door can be controlled via a real-time location, velocity and trajectory detection of a car. In this scenario, false-positive faults may lead to safety concerns. Another example might be the alarm system of a smart building deploying break beam/motion detection sensors. In this scenario, false-positive faults may cause unnecessary panic in the environment. Therefore, false-positive and false-negative faults must be addressed and handled to guarantee trustworthy and reliable operations of CPS applications using sensor networks.

The majority of related work focuses on techniques for sensor readings of non-binary measurements in control loops. Kalman filters [141] or simplex architectures [142] are popular approaches for such systems. However, such approaches do not address binary event detection and sporadic activity patterns.

A broadly used approach to increase robustness of sensor systems is modular redundancy [143], in which redundant sensors are deployed to detect and mitigate sensor faults. Triple modular redundancy (TMR) has been applied in many avionic systems [144]. In the area of modular redundancy, voting strategies [145], i.e., mechanisms to identify faulty sensor readings, are still an active field of research.

To reduce the need for a high number of redundant physical sensors, the concept of algorithmic redundancy has been proposed [146]. As an example, model-based prognostics [147] can be applied to cope with missing or faulty sensor information. In this case, domain-specific knowledge through the physical models or plant models are used to compensate for faults and predict the correct values.

Isermann [148] provides a survey on these approaches and shows their application in automotive industry. However, no single solution exists that suits all faults and system requirements in a typical application. This is caused by contradicting goals for the system, such as cost, mitigation of false-negative faults, mitigation of false-positive faults, or system performance. In addition, many of those timely event detection proposals in the earlier studies only consider timing uncertainties that come from the cyber side. However, the fundamental part of a CPS is the physical system. Therefore, a CPS application developer has to perform a detailed analysis on sensor fault semantics and timing attributes in the physical part, as well. For that kind of analysis, what is needed is a simulation framework that allows system designers to examine sensor faults and time-critical functionalities for event detection and manage design trade-offs in CPSs. To this end, we adopt a model-based design (MBD) methodology as introduced in Chapter 1 to develop our simulation framework.

3.3 Terminology and Fault Semantics

Before describing our model-based approach for reliable motion detection in our demonstrative example of a CPS, we need to clarify some terminology used throughout this chapter in order to make our discussion more clear. Since we aim to handle possible fault occurrences and temporal uncertainties during the event detection in a CPS, we need to clarify what an event, failure, fault, and timing uncertainty are.

An event is any detectable action or occurrence that is of importance to a system. CPSs are expected to reliably respond to the events both in the cyber and in the physical worlds. In the rest of this chapter, we will use the word event to refer to a physical phenomenon that occurs in the physical world. A binary event detector is a type of sensor which can return only a one-bit information in response to an occurrence of an event.

A failure is the lack of success in the outcome of a system. Since definition of success is application-specific, system failure and success are defined by the designer according to the mission to be realized.

A fault is an error that occurs in a component of a system. Faults on binary event detectors can be false-negative (FN) or false positive (FP). A false-negative fault is a type of error, which indicates that a binary event detector failed to detect an event even though the event occurred. A false-positive fault is a type of error which indicates that a binary event detector fired even though an event did not occur.

A timing uncertainty refers to an ill-defined or sporadic delay. Timing uncertainties are subtle causes for failure of a system. In many systems, they will be varying in a random fashion. In CPSs, they may be caused by the cyber or the physical system attributes.

In the real world, sensors do not fail frequently. Hence, faults are characterized statistically (i.e., using some probability model). Faults may occur over different domains (e.g., time

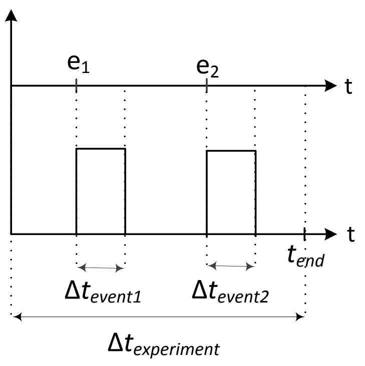


Figure 3.1: A sequence of events during one experiment.

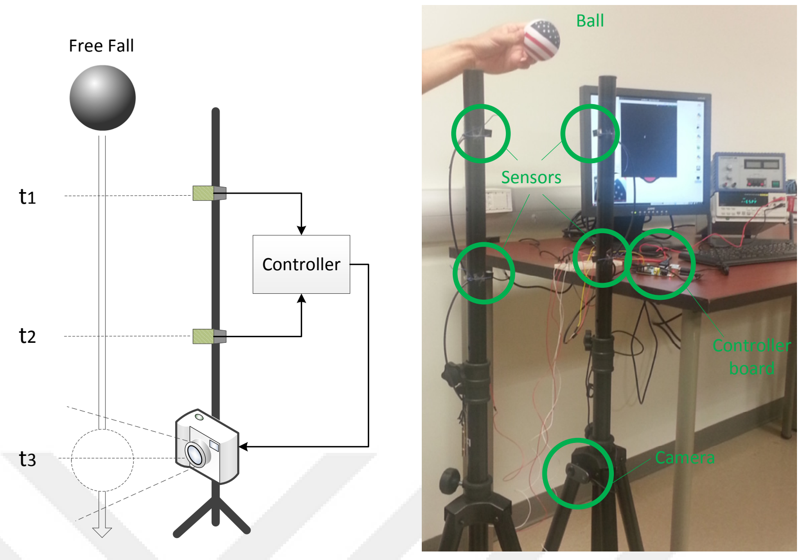
domain or event domain). In our study, false-negative and false-positive faults are measured over the time domain. The following equations define the occurrence probability of false-negative and false-positive faults, respectively:

$$P_{\mathrm{FN}} = \frac{\Delta faults_{\mathrm{FN}}}{\Delta t_{\mathrm{event}}} \tag{3.1}$$

Where P_{FN} refers to the probability of a false-negative fault, $\Delta faults_{\mathrm{FN}}$ refers to the number of false-negative faults, and $\Delta t_{\mathrm{event}}$ refers to the pulse width of event's detection and undetection.

$$P_{\mathrm{FP}} = \frac{\Delta faults_{\mathrm{FP}}}{\Delta t_{\mathrm{experiment}}} \tag{3.2}$$

Where P_{FP} refers to the probability of a false-positive fault, $\Delta faults_{\mathrm{FP}}$ refers to the number of false-positive faults, and $\Delta t_{\mathrm{experiment}}$ refers to a window of time when one iteration of the system operation is performed. Figure 3.1 shows $\Delta t_{\mathrm{event}}$ and $\Delta t_{\mathrm{experiment}}$, where e_1 and e_2 refer to subsequent events, $\Delta t_{\mathrm{event1}}$ and $\Delta t_{\mathrm{event2}}$ refer to the pulse width of event detection and undetection for event_1 and event_2, respectively, and t_{end} refers to the end of an iteration of the system operation.



(a) (b)

Figure 3.2: The falling ball example (FBE) baseline architecture a) Schematic view b) Experiment setup

3.4 A Demonstrative Example of a CPS

We investigate an instance of a uncomplicated CPS application, namely the falling ball example (FBE), suitable to be applied in the education of CPS engineers. The baseline architecture for FBE is shown in Figure 3.2, which includes two binary motion detectors (i.e., sensors), a camera (i.e., actuator), a controller, a ball, and necessary appliances for the setup. The controller interprets sensors' views of the movement of the ball, which is in a free-fall from a certain height, to estimate when it reaches the height of the camera in its trajectory. Then, a course of action is taken by the controller to trigger the camera. The camera takes a picture of the ball, when it reaches the height of the camera. The time for camera to act is predicted based on the information which comes from motion sensors mounted above the camera. The controller utilizes the feed-forward control strategy, which is regarded as anticipation of the relationship between the physical system and its environment and taking a necessary course of action according to this anticipation.

It is necessary to express the physical system as a part of the control algorithm with regular expressions. In case of a falling ball, we can apply the basic free fall equations. Using the known height of both sensors (h_1 and h_2), the value of gravity (g), and the event detection time of sensor$_1$ (t_1) and sensor$_2$ (t_2), we can compute the speed of the ball at sensor$_2$ (v_2) with the following equation:

$$v_2 = \frac{h_1 - h_2}{t_2 - t_1} + g\left(\frac{t_2 - t_1}{2}\right) \tag{3.3}$$

We can use v_2 to compute the expected time (t_3) of the falling ball to reach the height of camera (h_3) with the following equation:

$$t_3 = \frac{gt_2 - v_2 + \sqrt{{v_2}^2 + 2gh_2 - 2gh_3}}{g} \tag{3.4}$$

Computing Equation 3.4 is the most complex step (i.e., step 5) in the small control program, which is shown in Table 3.1.

Table 3.1: Control algorithm in FBE.

1. wait for a detect signal from sensor$_1$,
2. after receiving the signal from sensor$_1$, record the time (t_1),
3. wait for a detect signal from sensor$_2$,
4. after receiving the signal from sensor$_2$, record the time (t_2),
5. considering t_1 and t_2, compute the expected time (t_3) for actuator to act,
6. wait until t_3, then activate camera while compensating for the expected delay time in the system.

CPSs are supposed to deliver demanded services in timely and reliable fashion. Any failure of delivering these services is considered as a system failure. System failures may occur due to several reasons including but not limited to component faults, accuracy of sensor data, temporal uncertainties etc.

3.4.1 Understanding of Timing Uncertainties

Timing uncertainties depend on the system configuration. Each system needs to be analyzed for timing, separately. In order to find out the temporal behavior of the components in a lightweight CPS such as FBE, we analyzed the sensor response delay and the controller IO processing delay. In our system configuration, a Raspberry Pi board was used as a controller and a beam infrared sensor was used as an event detector. The followings are necessary equipment to realize timing measurements:

- Raspberry Pi model B revision 2.0 with Debian
- Linux and standard C development environment
- Beam infrared sensor (HOA6299 series), which detects a break of the light beam from the emitter to the detector
- 16800 series Portable Logic Analyzer
- 5 Mhz Function Generator

Figure 3.3 shows the setup schematic for timing measurements. The beam infrared sensor consists of two parts, namely the emitter which emits the light beam and the detector which detects the broken light beam. The function generator acts as a signal generator and triggers the emitter of the sensor at a certain frequency. The sensor only sends a detect signal to

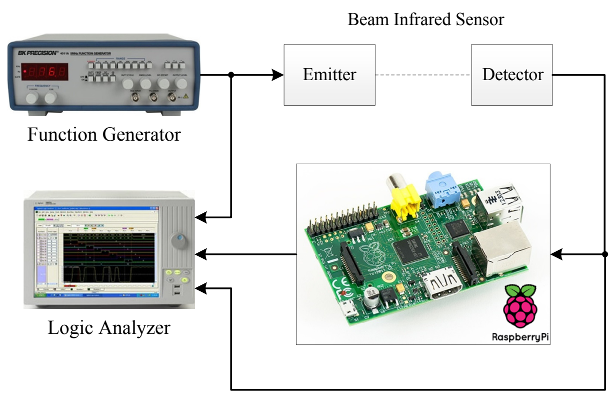


Figure 3.3: Timing measurement setup.

the controller (i.e., Raspberry Pi) when the light beam from the emitter to the detector is broken.

The controller can listen to the detector with two methods, namely the polling method and the interrupt method. The polling method refers to actively sampling the status of the detector by the controller. The controller does nothing other than checking the status of the detector until it is ready. The interrupt method refers to informing the controller

Table 3.2: Average controller IO processing delay.

Toggle Frequency (Hz)	**Average Delay (µs)**	
	Polling method	**Interrupt method**
1	15.2	80.4
10	12.2	53.8
50	10.7	45.3

Table 3.3: Average sensor response delay.

Toggle Frequency (Hz)	**Average Delay (ns)**
1	33.9
10	34.3
50	34.5

via a signal which indicates that an event has occurred. When the controller receives the interrupt signal, it takes an action specified by the interrupt service routine (ISR). The interrupt method allows the controller to engage in a different activity while waiting for the interrupt signal.

The logic analyzer was used for the exact measurement of the signal timings, namely sensor response and IO processing. It can record the timing with a precision of 4 ns. Table 3.2 shows the measurement results for the IO processing delay of the controller and Table 3.3 shows the results for the sensor response delay. The measurement results show that average IO processing delay is in the range of a few microseconds and average sensor response delay is in the range of a few nanoseconds. These delays are not likely to lead to a system failure. However, besides these delays, there may be subtle timing uncertainties from the physical side, which have a significant impact on the system outcome.

In the following section, we will address that in a scenario with our demonstrative example of a CPS in which the system success is degraded significantly because of timing uncertainties caused by the physical system attributes. We will compare the timing uncertainties from the cyber side, shown in Table 3.2 and Table 3.3, with the timing uncertainties from the physical side for that scenario.

3.5 Model-based Design of the CPS Example

The concept of model-based design (MBD) has been mentioned by various studies for efficient system development [149, 150, 151]. Model-based design allows the designer to design, analyze, verify and validate systems through several consequent design phases. It facilitates integrated system development and is a powerful design technique for CPSs [152].

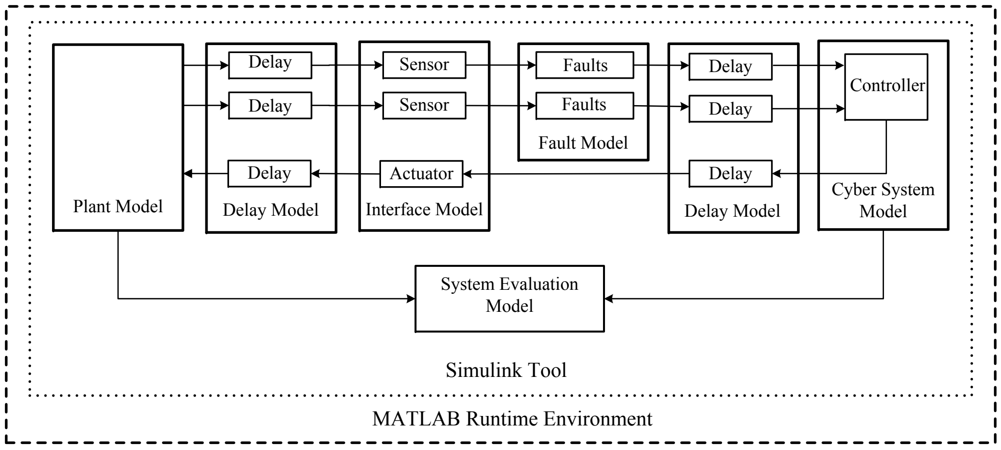


Figure 3.4: MATLAB/Simulink simulation framework.

3.5.1 A MATLAB/Simulink Simulation Framework

We use MATLAB/Simulink platform to implement our model-based design approach for CPS design. MATLAB is a high-level language and interactive environment for technical computing [153]. Simulink is a tool used to model and simulate the behavior of dynamic systems (those governed by differential equations) [154]. Simulink is integrated with MATLAB, enabling the designer to incorporate MATLAB algorithms into Simulink models and export simulation results to MATLAB for further analysis and visualization.

A typical CPS model consists of a physical system model, a cyber system model, and an interface model. We add fault models for each sensor and a system evaluation model into the typical CPS model in order to analyze effects of sensor faults on the system outcome and mitigate their adverse effects. A typical CPS model and new submodels added to it are shown in Figure 3.4. The whole simulation framework incorporates the physical system model, the interface model (including sensors and actuators), and the cyber system model, fault models, and the system evaluation model.

In our model-based development framework, the physical system model represents the physical phenomenon which needs to be monitored or controlled. The interface model represents

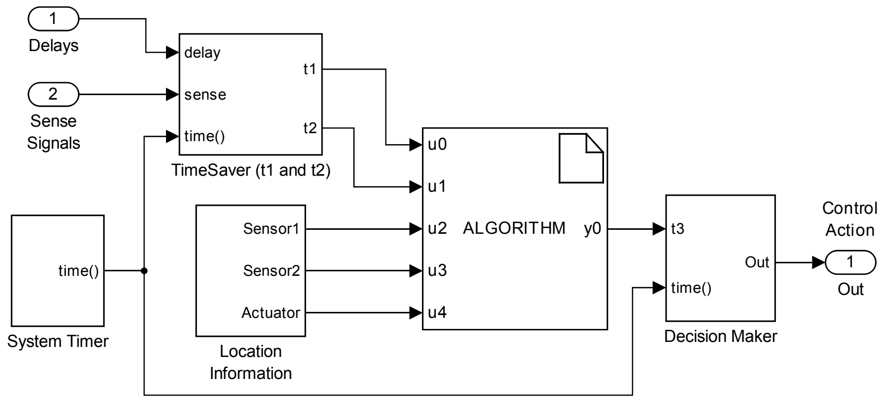


Figure 3.5: Controller subsystem.

the binary event detectors and actuators that are responsible for data gathering and control actions, respectively. The cyber system model comprises the controller and other logical units. The controller unit performs control computations and outputs control action according to the result. The interactions between the cyber and the physical systems are driven by events which occur in the physical world. The events are detected via binary event detectors and reflected in the cyber system to automatically make necessary control decisions. The control actions are taken by the cyber system as a result of timely event detection mechanism.

We capture the relationship between models in Figure 3.4 via Simulink [154] and our controller subsystem inside cyber system model is shown in Figure 3.5. The system timer unit is a counter that increment based on the sampling frequency as in microcontroller timers that increment based on the clock cycle. The timesaver unit is responsible for saving the time when the sense signal becomes true for a sensor. The timesaver provides t_1 and t_2 to the computing unit. The computing unit is responsible for processing the data provided by other units (e.g., t_1, t_2, and sensors' information). It uses step 5 in Table 3.1 to compute t_3. System time and t_3 are fed into the decision maker unit. The decision maker unit is responsible for taking necessary control action in due time (i.e., t_3).

With the model of a CPS as shown in Figure 3.4, our ultimate goal is to identify the relationship between sensor faults, timing uncertainties, and system success. We incorporated sensor fault generation and system evaluation functionalities into the FBE architecture in order to assess the system success under faulty sensor behaviors. Faulty sensor behaviors affect the success of the system adversely for sure. So that was the motivation for us to investigate fault mitigation techniques used to overcome adverse impacts of sensor faults and timing uncertainties on the system success.

Our framework can easily be extended to larger CPS applications using binary event detectors. The plant model, the interface model, and the cyber system model (i.e., control algorithm) may need modifications according to the system definition of the relevant CPS application since they are application-specific. However, without any modification, the fault generator model can be integrated into larger CPS simulation frameworks since binary event detectors show the same behavior (i.e., the provision of one-bit information in response to an occurrence of an event). Model-based design in Simulink facilitates this integration. The fault generator model should be placed between the sensor model and the cyber system model. The system evaluation model may need some modification as well since it considers inputs from the plant model and the cyber model for the system evaluation as shown in Figure 3.4.

3.5.2 Fault Model

The fault generator model shown in Figure 3.4 produces pre-defined faults for binary event detectors. We incorporated fault generation functionality into the CPS simulation framework to analyze the impact of false-negative and false-positive faults on the system outcome. Figure 3.6 shows the fault generator model comprising both the false-negative (FN) and false-positive (FP) fault generation functionalities.

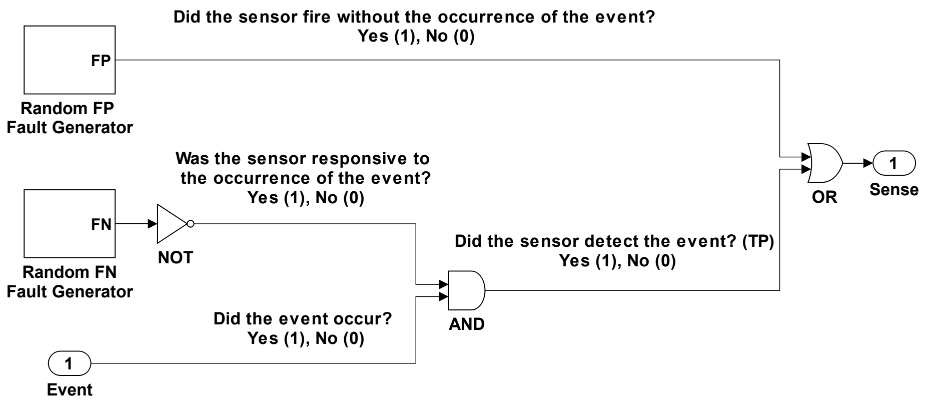

Did the sensor fire without the occurrence of the event? Yes (1), No (0)
FP
Random FP Fault Generator
Was the sensor responsive to the occurrence of the event? Yes (1), No (0)
FN
Random FN Fault Generator
NOT
Did the sensor detect the event? (TP) Yes (1), No (0)
AND
Did the event occur? Yes (1), No (0)
1
Event
OR
1
Sense

Figure 3.6: FP and FN fault generator model for single binary event detector.

A major challenge of the fault generator is the fact that a false-negative fault is meaningful only when the sensor is supposed to detect an event, while a false-positive fault can occur in any window of time. This fact was also expressed in Equation 3.1 and 3.2. We consider these equations to express the behavior, input, and output signals of the fault generator in the fault model.

Each binary $sensor_{\mathrm{i}}$ of the system has its individual fault generator. Each time the model is evaluated in Simulink, the following variables are computed:

$$FN_{\mathrm{i}} = \begin{cases} 1 & \text{if } \mathit{Event} \text{ is true and } \mathit{sensor}_{\mathrm{i}} \text{ doesn't fire;} \\ 0 & \text{otherwise.} \end{cases}$$

$$FP_{\mathrm{i}} = \begin{cases} 1 & \text{if } \mathit{Event} \text{ is not true and } \mathit{sensor}_{\mathrm{i}} \text{ fires;} \\ 0 & \text{otherwise.} \end{cases}$$

$$TP_{\mathrm{i}} = \begin{cases} 1 & \text{if } \mathit{Event} \text{ is true and } FN_{\mathrm{i}} \text{ is not true;} \\ 0 & \text{otherwise.} \end{cases}$$

Where FN_{i} indicates the occurrence of false-negative fault for $sensor_{\mathrm{i}}$, FP_{i} indicates the occurrence of false-positive fault for $sensor_{\mathrm{i}}$, and TP_{i} indicates the expected behavior of $sensor_{\mathrm{i}}$ when the physical phenomenon is in the sensing area of $sensor_{\mathrm{i}}$.

The above-mentioned variable formulations show the signal behaviors in the fault generator model as shown in Figure 3.6. The information regarding the event is provided to the fault generator model through the port *Event*. FN_{i} and FP_{i} are the outputs of random fault generator module. TP_{i} refers to the output of AND logical operator as shown in Figure 3.6. The fault generator model includes fault probabilities for false-negative and false-positive faults as a function of the model parameters.

3.5.3 Delay Model

Timing uncertainties (e.g., ill-defined or sporadic delays) may come from either sides, namely the cyber side or the physical side. Timing characteristics of the system components (e.g., sensor response delay, IO processing delay, computation delay, etc.) lead to timing uncertainties in the cyber system. The changes in inner dynamics (i.e., plant conditions) or outer dynamics (i.e., environment conditions) of the physical system may cause timing uncertainties and therefore CPSs may not be able to deliver demanded time-critical functions.

As mentioned in Subsection 3.4.1, what we have observed according to our timing measurements is that the delays which occur on the cyber side are very small. The sensor response delay is about a few nanoseconds and the IO processing delay of the controller is a few microseconds. However, there may be subtle timing uncertainties in the physical part of CPSs.

In our demonstrative example of a CPS, a significant timing uncertainty is caused by the misalignment of the ball. This misalignment is a part of the physical subsystem and causes

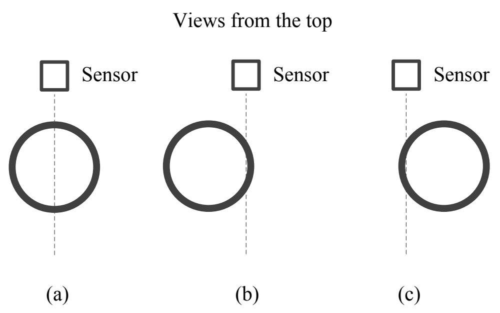


Figure 3.7: Horizontal alignment of the ball: A) Perfect alignment B) and C) Worst-case scenarios.

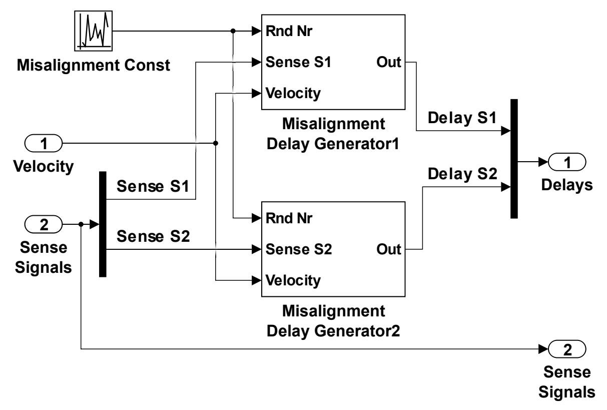
uncertainties in the range of milliseconds. With a few milliseconds delay, the system ends up with not delivering the time-critical functionality correctly. Figure 3.7 shows the scenarios related to the alignment of the ball in the experiment setup.

In the rest of this chapter, we will refer to the misalignment delay to address the delay due to any alignment of the ball other than the perfect one as shown in Figure 3.7. We assume every alignment of the ball is in the detection region of the sensors and since we drop the ball in free fall, we assume the ball drops in a straight path, vertically. Our delay model takes account of the misalignment delay. The misalignment delay is directly related to the velocity and the distance the falling ball takes, and therefore can be governed by the laws of physics as follows:

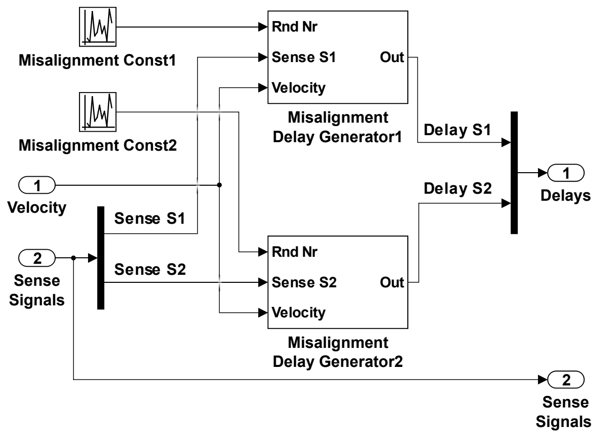
$$t_{\mathrm{m}} = \frac{d}{v} \tag{3.5}$$

where t_{m} is the delay due to the misalignment of the ball, d is the distance the falling ball takes, and v is the velocity of the falling ball. The following constraint is considered for the distance d.

$$\{d \in R \mid 0 \leq d \leq r\} \;\; and \;\; r = 3 \;\; cm$$



(a)



(b)

Figure 3.8: Misalignment delay generation for timing analysis a) Vertical b) Horizontal.

where r is the radius of the ball. The following constraints are used to calculate the event detection time of sensor$_1$ and sensor$_2$ (i.e., $t_1{}'$ and $t_2{}'$), respectively, considering the misalignment delay.

$$\left\{ t_1{}' \in R \mid t_1 \leq t_1{}' \leq \left(t_1 + \frac{d}{v_1} \right) \right\}$$

$$\left\{ t_2{}' \in R \mid t_2 \leq t_2{}' \leq \left(t_2 + \frac{d}{v_2} \right) \right\}$$

where $t_1{}'$ and $t_2{}'$ refer to the event detection time of sensor$_1$ and sensor$_2$ considering the misalignment delay. t_1 and t_2 refer to the exact starting time of the event for sensor$_1$ and sensor$_2$, respectively. v_1 and v_2 refer to the velocity of the falling ball when it reaches the height of sensor$_1$ and sensor$_2$, respectively.

We apply the aforementioned equations and constrains to our delay model for the misalignment delay. Figure 3.8a shows the misalignment delay generators in the delay model with vertically aligned sensors. Since the sensors are aligned in the same straight path vertically in the FBE baseline architecture, the same uniform random number generator is used to define a misalignment constant for either sensors. In the case where sensors are aligned

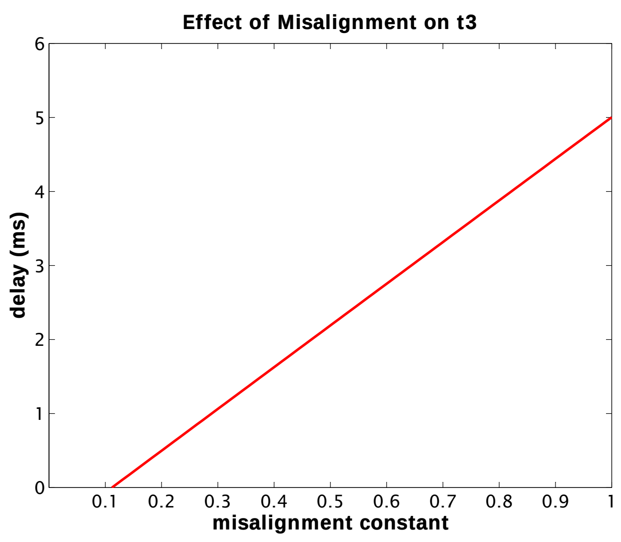


Figure 3.9: The effect of ball misalignment on the calculated time of the control action (i.e., t_3).

horizontally, different misalignment constant must be defined for each sensor in the same horizontal line as shown in Figure 3.8b. The misalignment constant ranges between 0 and 1. In the perfect case, the constant is 0 and in the worst case, the misalignment constant is 1. The delay value is calculated according to the aforementioned formulas and constraints.

Figure 3.9 shows the effect of ball misalignment on the calculated time of the control action (i.e., t_3) when we assume that the ball is dropped with the free-fall acceleration from a certain height (i.e., 2m) and the heights of sensor$_1$ (h$_1$), sensor$_2$ (h$_2$), and camera (h$_3$) are specified as 1.5m, 1m, 0.5m, respectively. In the worst case scenario, it leads to approximately 5 ms delay added to t_3 which is likely to prevent the control action from being taken correctly. Compared to the IO processing delay of the controller and the sensor response delay, as shown in Table 3.2 and Table 3.3, respectively, the delay caused by ball misalignment has much greater impact on the system outcome. Therefore, it needs to be handled to improve the system success.

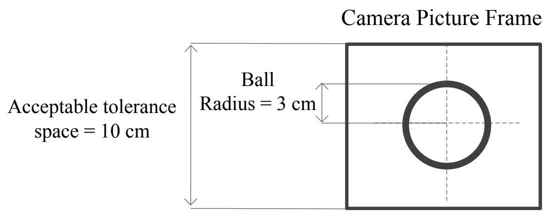


Figure 3.10: System success criteria.

3.5.4 System Evaluation Model

Since the definition of system success is application-specific, system failure and success are defined by the designer according to the mission to be realized. In FBE, the goal is to take a snapshot of the falling ball in a timely manner and the definition of success for FBE is that the falling ball should be in the middle of the snapshot with an acceptable tolerance space. Success rate for FBE can be defined as follows:

$$Success\ Rate = \frac{Number\ of\ Successful\ Shots}{Total\ Number\ of\ Shots} * 100 \tag{3.6}$$

A successful shot denotes the placement of the ball in the center of the camera picture frame with a certain tolerance as shown in Figure 3.10.

The acceptable tolerance space is 10 cm in the frame of a snapshot and can be formulated as follows:

$$T = 2 * (r + c) \tag{3.7}$$

where T refers to the accepted tolerance space, r refers to the ball radius and c refers to the tolerance constant which is 2 cm. The success is specified as the whole ball in the frame of the camera with a tolerance of 2 cm at the time when it reaches the height of the camera. Then the success rate is calculated according to Equation 3.6.

The system evaluation model assesses the accuracy/satisfaction of the system outcome after the control action. The control signal and information from the physical system is fed into the model. The system evaluation model performs comparison and evaluates correct functionalities. Success of the system is evaluated according to the definition of system success provided by the system designer. This definition is usually application-specific. In FBE, the success rate decrease significantly with the injection of faults and delays into the CPS model. The following section deals with how to increase success rate under the occurrences of fault and delay.

3.6 Fault Mitigation Techniques

Time-critical functions may not be delivered in case sensor faults and subtle delays occur. Since CPSs are expected to be highly reliable, the CPS designer should be clear about the following question:

- Is it possible to ensure reliable behavior under fault occurrence and timing uncertainty?

Certain fault and delay handling mechanisms need to be applied to ensure reliability of the system. A variety of fault mitigation techniques have been proposed, ranging from component redundancy [143], over algorithm redundancy [147] to complex mathematical models [142]. However, the choice of technique is likely to depend on the application domain since no single solution fits each application in various domains. When the redundancy approaches are considered, the application developer has to perform an analysis on trade-offs between cost, design time, type of faults and delays that need to be handled.

Possible solutions, which spring to mind in order to handle potential false-negative fault, false-positive fault, and timing uncertainties in our demonstrative example of CPS, are sensor and control algorithm redundancies.

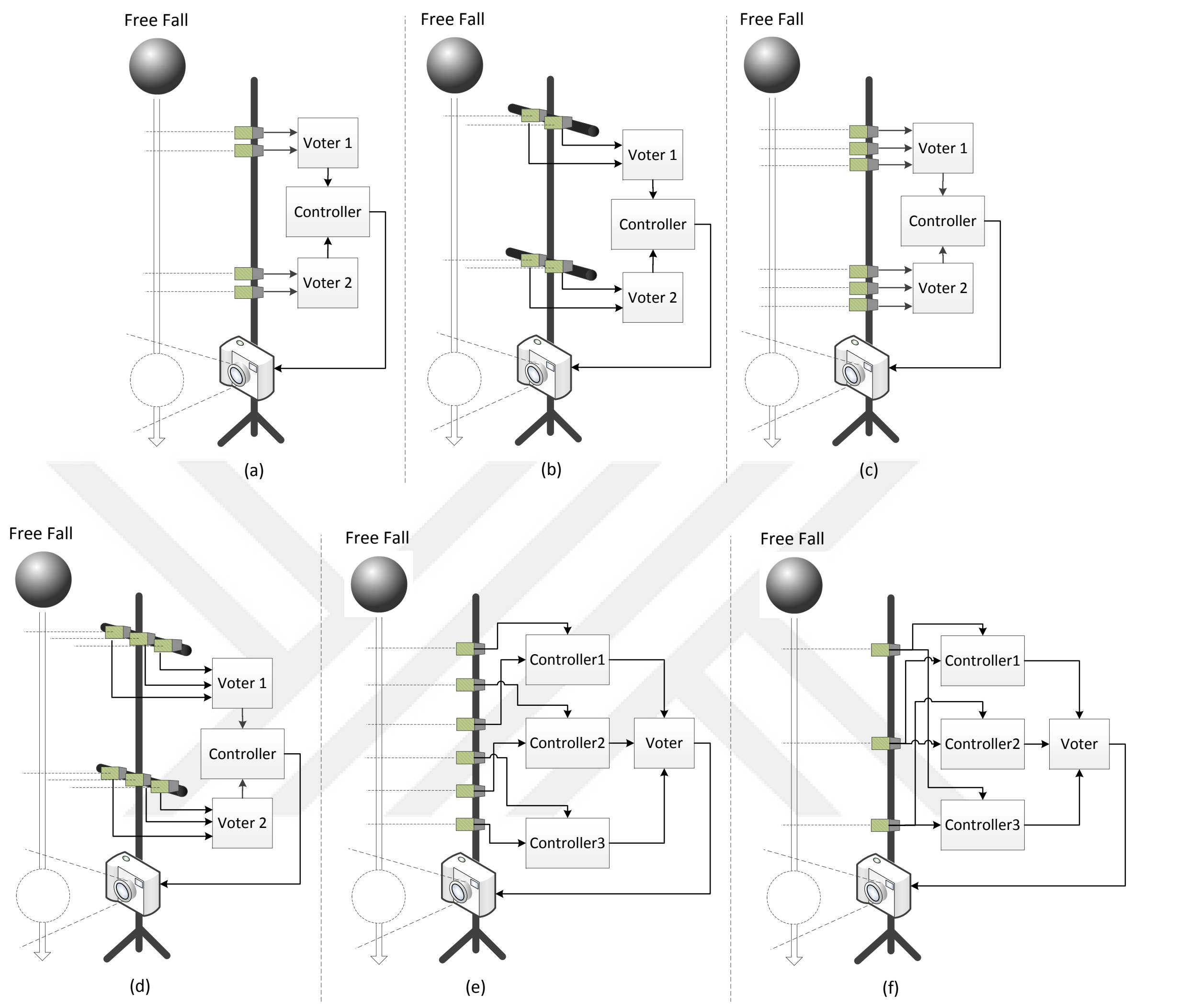


Figure 3.11: Fault and delay handling techniques for FBE (a) VSR4, (b) HSR4, (c) VSR6, (d) HSR6, (e) TCR6, (f) TCR3.

We propose the following architectures as shown in Figure 3.11,

- Dual vertical sensor redundancy (denoted as VSR4)
- Dual horizontal sensor redundancy (denoted as HSR4)
- Triple vertical sensor redundancy (denoted as VSR6)
- Triple horizontal sensor redundancy (denoted as HSR6)
- Triple controller redundancy with three vertically aligned sensors (denoted as TCR3)
- Triple controller redundancy with six vertically aligned sensors (denoted as TCR6)

which are defined according to various choices (e.g., sensor locations, number of sensors, number of control algorithms, etc.). Our aim is to explore the design space of our demonstrative CPS example and find out the suboptimal design decisions according to the fault types and delay conditions. The following subsections deal with the classification of the proposed architectures.

3.6.1 Component Redundancy

As a solution to those component faults, namely false-negative and false-positive faults, the traditional approach is to apply sensor redundancy. Agreement of sensors on the presence of the physical phenomenon will prevent sensor component faults (i.e., false-positive and false-negative) to affect the system outcome adversely.

3.6.1.1 Vertical Sensor Redundancy

Sensors may be used redundantly in the same alignment vertically for more reliable operation. Figure 3.11a and 3.11c show dual and triple vertical sensor redundancy approaches. They are denoted as VSR4 and VSR6, respectively.

VSR4 comprises four vertically aligned sensors, two voters, and one controller. Each voter is responsible for applying selection logic (i.e., OR) on the sensor signals and applies logical OR operation to both sensor signals. An OR operation means that if only one sensor detects the event, that's enough. Each voter also considers delays (e.g., sensor response delay, IO processing delay, and ball misalignment delay, etc.) and selects the sense signal with lowest delay.

VSR6 comprises six vertically aligned sensors, two voters, and one controller. The voter mechanisms ensure that at least two of three sensors agree on the sense signals. Each voter also considers delays (e.g., sensor response delay, IO processing delay, and ball misalignment delay, etc.) and selects the sense signal with lowest delay. The agreement of sensors on the occurrence of the event will prevent sensor component faults (especially false-positive faults) to affect the system outcome, adversely.

In both architectures, the sense signal will be forwarded to the subsequent system units. The output of voter$_1$ represents t$_1$ and the output of voter$_2$ represents t$_2$, the falling ball reaches the height of the camera at t$_3$.

3.6.1.2 Horizontal Sensor Redundancy

Sensors may be used redundantly in the same alignment horizontally. Figure 3.11b and 3.11d show dual and triple horizontal sensor redundancy approaches. They are denoted as HSR4 and HSR6, respectively.

HSR4 comprises four sensors, two voters, and one controller. At a certain height, there are two horizontally aligned sensors, which provide detection information to the voter$_1$, another two horizontally aligned sensors, which provides detection information to the voter$_2$. The controller is responsible for taking necessary control action. A voter here works as described in VSR4. The difference between VSR4 and HSR4 comes from the delay model. While same misalignment constant is used in VSR4 for all sensors to generate misalignment delay, different misalignment constant is used in HSR4 for each sensor in the same horizontal line.

HSR6 comprises six sensors, two voters, and one controller. At a certain height, there are three horizontally aligned sensors, which provide detection information to the voter$_1$, another three horizontally aligned sensors, which provides detection information to the voter$_2$, and the controller is responsible for taking control action. Each voter works in the same way described for VSR6. The difference between VSR6 and HSR6 comes from the delay model. While same misalignment constant is used in VSR6 for all sensors to generate misalignment delay, different misalignment constant is used in HSR6 for each sensor in the same horizontal line.

In both architectures, the sense signal will be forwarded to the subsequent system units. The output of voter_1 represents t_1 and the output of voter_2 represents t_2, the falling ball reaches the height of the camera at t_3.

3.6.2 Algorithm Redundancy

Control algorithms may be used redundantly to handle fault and timing uncertainties. Figure 3.11e and 3.11f show triple control algorithm redundancy with the use of different number of sensors. The triple control algorithm with six sensors is denoted as TCR6. It comprises six sensors, one voter, and three controllers as shown in Figure 3.11e. The triple control algorithm redundancy with three sensors is denoted as TCR3. It comprises three sensors, one voter, and three controllers as shown in Figure 3.11f.

Both architectures deploy three controller units in the cyber part. Each controller waits for data from different sensors located in the trajectory of the falling ball. The output of each controller represents an instance of t_3. The voter mechanism ensures that at least two of three controllers agree on the value of t_3 with a tolerance of 10ms. This tolerance reflects different misalignment delays of sensors due to their placement in the system setup. When at least two controllers provide the values of t_3 with at most 10 ms difference, then the smallest one of those instances of t_3 is considered to be the expected value of t_3.

Table 3.4: Simulink model configurations.

Simulation time (t_{sim})	2 sec
Simulation time of the plant (t_{plant})	0.6 sec
Simulation start time for the plant (t_{start})	random number that ranges from 0 to 1.4 sec (t_{sim} - t_{plant})
Simulation stop time for the plant (t_{stop})	$t_{\text{start}} + t_{\text{plant}}$
Sampling time of FN fault	2 sec
The occurrence probability of FN fault	ranges from 0 to 0.05 with step size of 0.005
Sampling time of FP fault	200 msec
The occurrence probability of FP fault	ranges from 0 to 0.05 with step size of 0.005

3.7 Experiments And Evaluation

We developed the models shown in Figure 3.4 for the architectures represented in Figure 3.11 via Simulink to evaluate the system reliability and satisfaction of timely system response. In the system simulation, we follow an episode scenario where the falling ball activity occurs randomly during a window of time (i.e., $\Delta t_{\text{experiment}}$ as previously mentioned in Section 3.3). That window of time is two seconds. We defined false-negative fault as a random binary number generated at the time interval of an event (i.e., Δt_{event}). The time interval of Δt_{event} changes according to the velocity of the falling ball at certain height. The sampling time of false-negative fault is one millisecond. We defined false-positive fault as a random binary number generated in a certain sampling time (i.e., two hundred milliseconds) at the time interval of $\Delta t_{\text{experiment}}$. We considered a fault probability range from 0 to 5%. Since false-positive and false-positive faults are generated in the given probability range and sampling time, the work is left to the system evaluation model to find out correct results. The Simulink model configuration parameters for the simulation are summarized in Table 3.4.

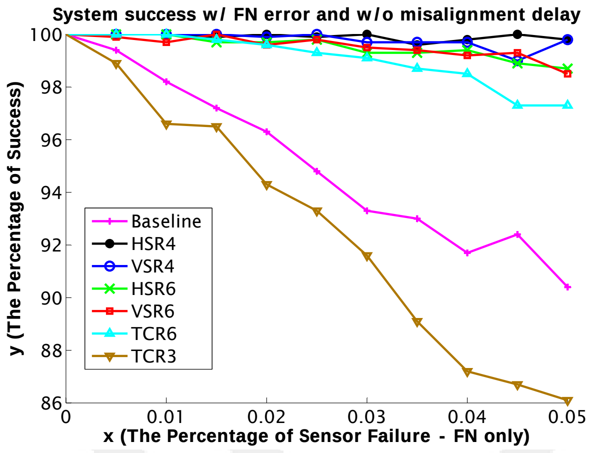
3.7.1 Simulation Results

We compared seven architectures, namely baseline architecture, dual vertical and horizontal sensor redundancy with OR logic denoted as VSR4 and HSR4, respectively, triple vertical and horizontal sensor redundancy with majority voting logic denoted as VSR6 and HSR6, respectively, triple control algorithm redundancy with six sensors and with three sensors denoted as TCR6 and TCR3, respectively. In our experiments, our aim is to find out the better approach for achieving more dependable and reliable system outcome.

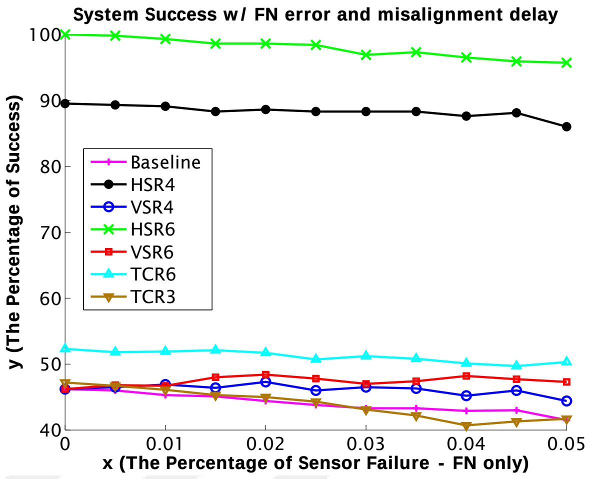
Figure 3.12 shows the system success rates for the given fault probabilities when not considering misalignment delay (3.12a, 3.12c, 3.12e) and when considering misalignment delay (3.12b, 3.12d, 3.12f) for the proposed architectures. In Figure 3.12, the x-axis represents the fault occurrence probability for false-negative (FN) and/or false-positive (FP) faults, and the y-axis represents the system success rate as formulated in Equation 3.6.

Figure 3.12a shows the system success rate in case only false negative fault occurred without taking account of misalignment delay. VSR4 and HSR4 show better performance than others in this category. They provide about 10% improvement in 5% fault probability (only FN) compare to the baseline architecture.

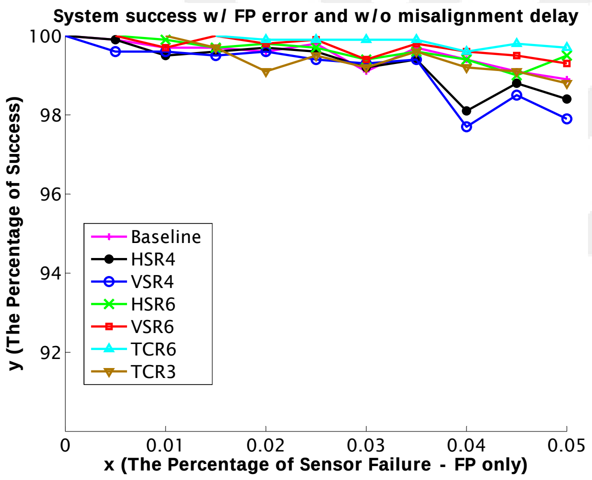
Figure 3.12b shows the system success rate in case only false negative fault occurred while considering misalignment delay. HSR6 shows the best performance in this category. It provides about 55% improvement in 5% fault probability (only FN) compare to the baseline architecture.



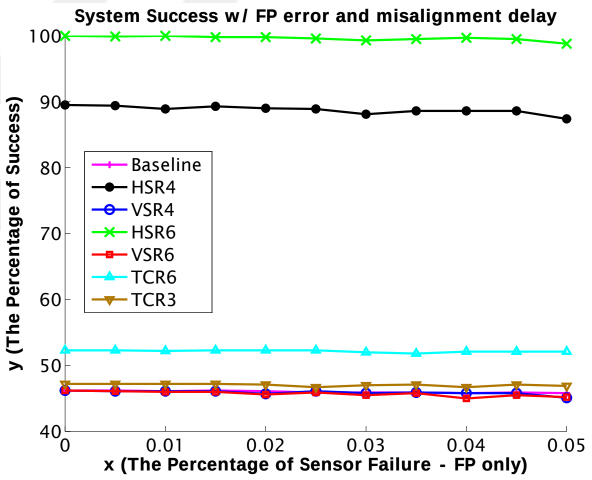
(a)



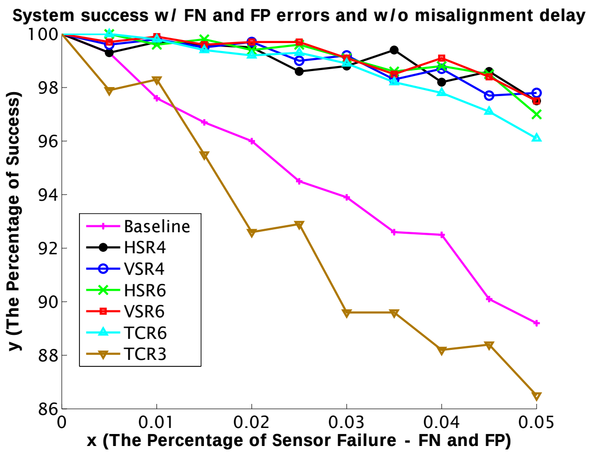
(b)



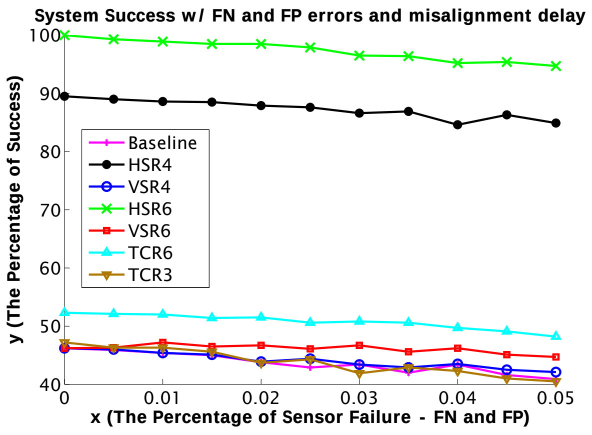
(c)



(d)



(e)



(f)

Figure 3.12: Success rate of different architectures with: FN only (a) Without delay, (b) With delay; FP only (c) Without delay, (d)With delay; FP only e)Without delay, f)With delay; FN and FP together.

Figure 3.12c shows the system success rate in case only false-positive fault occurred without taking account of the misalignment delay. TCR6 shows the best performance in this category. It provides about 2% improvement in 5% fault probability (only FP) compare to the baseline architecture.

Figure 3.12d shows the system success rate in case only false-positive fault occurred while considering misalignment delay. HSR6 shows the best performance in this category. It provides about 50% improvement in 5% fault probability (only FP) compare to the baseline architecture.

Figure 3.12e shows the system success rate in case false-positive and false-negative faults are taken into account together without taking account of misalignment delay. In this category, VSR4, HSR4, VSR6, and HSR6 show better performance than the others. They provide about 9% percent improvement in 5% fault probability (FN and FP together) compare to the baseline architecture. Each one of FN and FP has 5% occurrence probability.

Figure 3.12f shows the system success rate in case false-positive fault, false-negative fault, and misalignment delay are taken into account. Again, HSR6 shows the best performance in this category. It provides about 55% improvement in 5% fault probability (FN, FP, and misalignment delay together) compare to the baseline architecture.

As shown in figures 3.12b, 3.12d, and 3.12f, the success rate of architectures with vertically aligned sensors significantly decrease with the inclusion of misalignment delay on the system model. According to our results, we can conclude that some architectures might be better choices only for FN while some others might be better choices only for FP. However, timing uncertainties from either sides of the CPS might really have a serious adverse impact on the system behavior. Design space of a CPS application must be carefully explored by considering both the cyber and physical system attributes. Trade-offs need to be considered by the designer to fulfill stringent requirements of CPSs.

3.7.2 Design Space Exploration

The data presented so far shows the general variability of the effect of fault mitigation techniques in CPS. However, system success is not the only metric that is important in the design of CPS. Besides system success, different factors such as cost, energy consumption, fault tolerance, etc. need to be considered for multi-objective design space explorations. In this subsection, we explore the design space of our CPS use case application under the four main criteria, namely cost, FN, FP, and delay. FN, FP, and delay metrics are derived from the experiment results in the previous section. The system success is evaluated according to Figure 3.12. Cost metric is derived from the number and price of sensors and controllers. Table 3.5 shows the ranking of architectures based on those metrics. In the ranking, 1 indicates the best choice and 7 indicates the worst choice. According to that ranking, we draw the plots in Figure 3.13 in order to find out Pareto-optimal solutions in the design space. An ideal system would be in the lower left corner. We will see that the result of design space exploration differs depending on our system goal.

We conduct a Pareto analysis to rule out system setups that cost more and provide less system success. Figures 3.13a, 3.13b, and 3.13c show a Pareto analysis of design space

Table 3.5: Ranking of the architectures according to four criteria (cost, FN, FP, and delay).

Architecture	**Cost**	**Only FN**	**Only FP**	**FN + FP**	**FN + FP + Delay**
Baseline	1	6	4	6	7
VSR4	3	2	7	1	5
HSR4	4	1	6	3	2
VSR6	5	4	2	2	4
HSR6	6	3	3	4	1
TCR3	2	7	5	7	6
TCR6	7	5	1	5	3

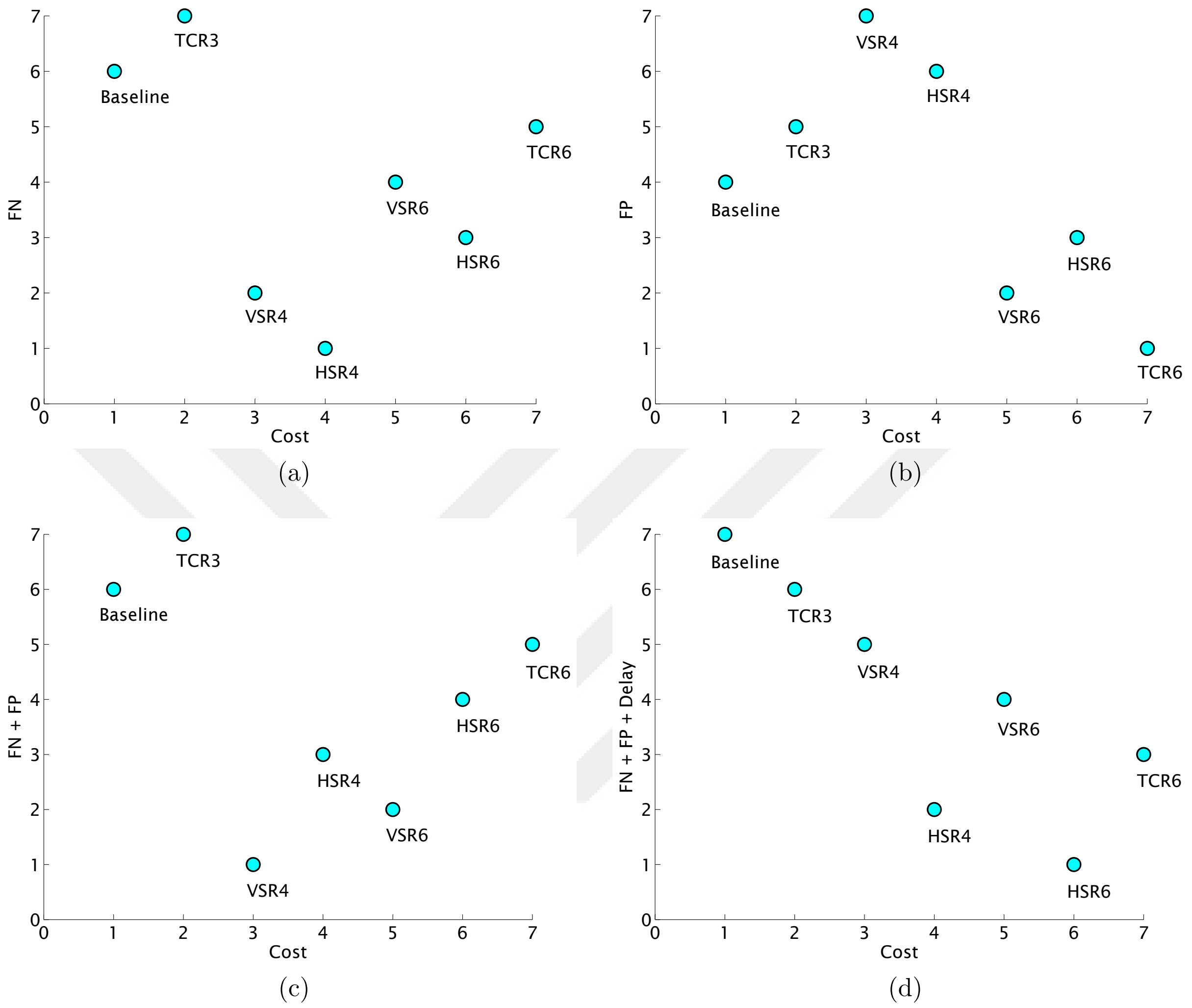


Figure 3.13: Design space of our CPS use case application: a) Cost vs false-negative trade-off, b) Cost vs false-positive trade-off, c) Cost vs false-negative and false-positive trade-off, c) Cost vs false-negative, false-positive, and delay trade-off

exploration for our CPS application in case of considering only FN fault and cost, only FP fault and cost, FN&FP faults and cost, respectively. Figure 3.13d shows a Pareto analysis of design space exploration for our CPS application in case sensor faults, timing uncertainties, and cost are considered. The choice of design after exploration changes depending on our system goal. Indeed, if we change the system parameters or objectives, the result of design space exploration will change. As our results indicate, this change is partly non-intuitive and non-trivial. Initially, we expected a direct trade-off between cost and system success. The results, however, show that we can remove about 60% of the design options. These results underline the importance of an evaluation framework which follows a model-based design approach highlighted in Figure 1.1 as it has been proposed in Chapter 1.

3.8 Conclusions

In this chapter, we addressed fulfilling timely event detections and delivering time-critical operations reliably in CPS under sensor fault occurrences and timing uncertainties (e.g., ill-defined or sporadic delays). We adopted a model-based design approach and injected well-defined sensor fault and delay model to the traditional CPS model that comprises the cyber, the interface and the physical model. Our measurements regarding timing analysis show that the physical system attributes (e.g., sensor placement and environmental effects) might be more dominant than the cyber system attributes, hence causing timing uncertainties and degrading CPS reliability. If not examined thoroughly, these subtle timing uncertainties might decrease the system success significantly when combined with the adverse impact of sensor faults on the system outcome.

We proposed a number of architectures which exploit sensor and control algorithm redundancies to mitigate the sensor faults and timing errors. Our results show that large delay implies reduced reliability region on the system operation. The CPS designer must consider not only the number of the sensors but also the placement of sensors as an important factor for design decisions on reliable CPS. We also outlined the design space of a fault-tolerant CPS application to ensure reliable system operation. Our results indicate that even for small CPS applications, the design space is complex and the best system setup is non-trivial, in particular when taking into account aspects such as cost or required precision for timing and fault-tolerance. We could demonstrate the effectiveness of our framework to simulate the effects of faults and timing uncertainties on the system outcome by means of the described use case and the result of design space exploration analysis.

Chapter 4

Sensor Fault Modeling and Mitigation for Smart Building as a CPS

4.1 Introduction

CPS technologies touch every aspect of our lives. Since most people spend up to 90% of their time indoors [155], building management/automation, also known as the smart building, plays an important role in everyday lives of people as a CPS application. Various sensors can be deployed in smart buildings to extract data from physical phenomena such as air temperature and quality. With the help of the data extracted, control of a building's heating, ventilation, and air conditioning (HVAC) systems can be achieved through a centralized decision making mechanism. If, however, we assume that the building has multiple rooms and occupants of each room have different expectations when it comes to the conception of comfort, then the centralized control approach for HVAC systems may fail to meet expectations of all occupants. Some occupants may be fully satisfied with ambient conditions inside the room while some others may feel unsatisfied with those conditions.

As a matter of fact, a survey conducted by International Facility Management Association (IFMA) in 2009 shows common HVAC complaints [156]. Received throughout the year, 94% of common HVAC complaints is regarding ambient temperature being too cold and 91% of those complaints is regarding ambient temperature being too hot. Indoor air quality complaint accounts for 25% of common HVAC complaints. Among HVAC complaints received in the summer, under air conditioned (i.e., too hot) complaint accounts for 66% of the complaints and over air conditioned (i.e., too cold) complaint accounts for 58% of the complaints. Seasonally speaking, too hot complaint is the most common one for the summer while too cold complaint is the most common one for the winter. Since outdoor temperature expectedly fluctuates in the spring and fall seasons, the results show that HVAC system can not adapt to those temperature changes as fast as expected and the complaints during these seasons are evenly distributed regarding too hot or too cold.

Besides HVAC related ineffectiveness, indoor air condition may be further aggravated by sensor faults since reliable HVAC system behavior depends on the data extracted from the phenomenon of interest (i.e., air temperature). These faults may lead to discomfort for occupants and energy consumption more than estimated. IFMA survey [156] shows that the occupants adjust to thermal comfort issues by obtaining personal fans or heaters, wearing supplemental clothes, blocking or redirecting vents, or adjusting sensors. These remedies may lead to rising need of in-building maintenance, increasing energy consumption, and unsafe conditions (e.g., risks of having personal heaters) in the buildings.

Distributed control of a building's HVAC systems provides great flexibility to ensure high quality residential comfort and prevent HVAC ineffectiveness in multi-room smart buildings. Moreover, sensor fault mitigation strategies deployed on a multi-room building help enhance the overall HVAC system effectiveness. Therefore, distributed control of a building's HVAC systems and sensor fault mitigation techniques must be made optimum use of in order to bridge the gap between occupants' expectations and their perceptions of a smart building.

In this chapter, we investigate temperature sensor faults and their impacts on the thermal comfort of occupants and HVAC energy efficiency.

The rest of this chapter is organized as follows. Related work and motivation are discussed in Section 4.2. Terminology and fault semantics are introduced in Section 4.3. A demonstrative example of CPS (i.e., multi-room smart building) is introduced in Section 4.4. Model-based design of the demonstrative example is reviewed in Section 4.5. Proposed fault mitigation techniques are discussed in Section 4.6. Experiments and system evaluation are discussed in Section 4.7. Finally, the chapter is concluded with Section 4.8. A portion of this chapter was published in [3].

4.2 Related Work and Motivation

CPS technology enables creating smart living spaces such as smart buildings of the future. Thermal comfort of occupants is an important aspect in the automation of smart buildings. Temperature sensor faults may cause occupants' dissatisfaction with indoor ambient conditions as well as energy inefficiencies. Various authors have studied fault detection mechanisms under fault occurrences in the buildings with HVAC systems.

The design of a model-based methodology is studied in [157] in order to detect and isolate multiple sensor faults in multi-zone HVAC systems where the interconnected subsystems are characterized by heterogeneous nonlinear dynamics. The authors propose a model-based and distributed architecture for sensor fault detection, which relies on robust analytical redundancy relations.

A system-level fault detection and diagnosis (FDD) strategy is proposed in [158] for HVAC systems involving sensor faults at the system level. The authors conduct analyses regarding the impact of system faults on the sensor FDD and the impact of corrected sensor faults on the system FDD. They point out that a component-level FDD method is recommended to figure out the root cause of performance degradation in the system.

A fault detection and diagnosis strategy is proposed in [159] using combined neural networks and subtractive clustering analysis in order to improve energy efficiency and thermal comfort in the buildings. The strategy is aimed at detecting faults such as soft sensor faults (e.g., fixed biases and drifting biases), complete failure of the sensors, and chilled water valve faults in the supply air temperature control loop of HVAC system and removing those faults.

A hybrid model-based fault detection technique is proposed in [160] by combining the classical fault detection method based on statistical residual evaluation with the one based on fractal correlation dimension (FCD) algorithm. The hybrid technique is validated through the transient system simulation tool (TRNSYS), considering six fixed bias faults injected into temperature sensor model under different load conditions during summer season. The authors conclude that the fixed bias faults can be detected efficiently by the hybrid technique.

Given the need for thorough understanding of sensor faults' semantics and their effects on the comfort of occupants, we investigate sensor faults observed in real world deployments and consider a multi-room smart building as a case study. The following section deals with sensor fault semantics and introduces the terminology used throughout the chapter.

4.3 Terminology and Fault Semantics

Modern buildings are comprised of several components such as HVAC systems, sensors, actuators, controllers, and so on. It is inevitable that some of these components may fail over time and have an adverse affect on the system outcome. Fault semantics specifies the behavior of a CPS under the occurrence of faults. In this study, we specifically focus on sensor faults. Therefore, in this section, we provide the taxonomy of faults for temperature sensors and clarify some terminology used throughout the chapter in order to make our discussion more clear.

Throughout this chapter, a sensor fault refers to a sensor's reading that is inconsistent with the expected behavior of the physical phenomenon. In order to systematically define the faults for temperature sensors, we examine the semantics of a number of sensor data faults observed in the real-world sensor deployments [161, 162]. The type of faults we analyze are as follows:

- single-sample spike fault (positive and negative spikes)
- spike-and-stay fault (positive and negative spikes)
- stuck-at fault

Single-sample spike fault refers to a sharp change that randomly occurs on only single point of the sampled values. This spike may take a positive direction, meaning a sharp increase in the sensor reading, or a negative direction, meaning a sharp decrease in the sensor reading. Spike-and-stay fault refers to a sharp change which occurs randomly and subsequently preserves its value for a large number of consecutive points of the sampled values. Stuck-at fault refers to a range of sensor readings that undergo zero rate of change. We assume that the sensors having these faults return to the correct behavior afterwards.

We developed well-defined models of these faults and injected those fault models to the CPS model for our demonstrative example, which is a multi-room smart building, in accordance with our model-based design (MBD) methodology as addressed in Chapter 1. We implemented fault generators in Simulink for each type of sensor fault mentioned above. Later in Section 4.5.2, we provide the illustrative plots from Simulink models for the above-mentioned faults to make the fault semantics more clear.

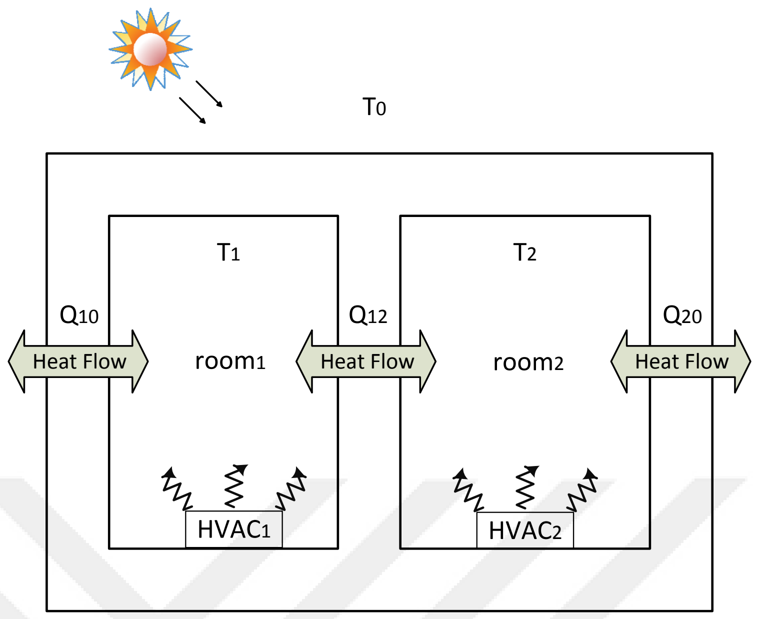


Figure 4.1: Heat flow and geometry of the building having two adjacent rooms.

4.4 A Demonstrative Example of the CPS

We investigate a multi-room building integrating temperature sensors, controllers, and HVAC systems as a CPS application. The phenomenon being observed is the room temperature. Our goal is to control room temperatures in a distributed fashion and meet expectations of all occupants with the emphasis on thermal comfort and energy consumption. A temperature sensor and controller are deployed in every room. Each room is provided with an HVAC system. The controllers utilize the data extracted via temperature sensors to turn on/off HVAC systems.

The building and HVAC systems constitute the physical part of our demonstrative example of a CPS, therefore it is very important for us to have well-defined properties of the building and HVAC systems. Since we observe the temperature variation in the building, thermal model of the building and HVAC systems must be well-defined as well. Figure 4.1 shows a sample 2-room building as an illustration of heat flows in the building. T_1 and T_2 refer to room temperatures for room$_1$ and room$_2$, respectively. T_0 refers to outside temperature.

Q_{10} represents the heat flow from room$_1$ to outside. Q_{20} represents the heat flow from room$_2$ to outside. Q_{12} represents the heat flow from room$_1$ to room$_2$.

We describe the thermal behavior of a multi-room building through the laws of thermodynamics. The thermal behavior of entire system is captured by thermodynamic equations [163] which consist of differential and algebraic equations as follows:

$$C_1 \frac{d}{dt} T_1(t) = Q_{\mathrm{H1}} - Q_{10} - Q_{12} \tag{4.1}$$

$$C_2 \frac{d}{dt} T_2(t) = Q_{\mathrm{H2}} - Q_{20} + Q_{12} \tag{4.2}$$

$$Q_{10} = \frac{T_1(t) - T_0(t)}{R_{10}} \tag{4.3}$$

$$Q_{20} = \frac{T_2(t) - T_0(t)}{R_{20}} \tag{4.4}$$

$$Q_{12} = \frac{T_1(t) - T_2(t)}{R_{12}} \tag{4.5}$$

where C_1 and C_2 refer to the thermal capacitance parameters for room$_1$ and room$_2$, respectively. T_0, T_1, and T_2 refer to temperatures for outside, room$_1$, and room$_2$, respectively. R_{10}, R_{20}, and R_{12} refer to thermal resistance parameters for room$_1$ and room$_2$. Q_{10}, Q_{20}, Q_{12} refer to heat flows. Q_{H1} and Q_{H2} refer to hot air supplies provided by HVAC$_1$ and HVAC$_2$ systems, respectively.

Same equations are used to model the effect of cool air supplies provided by HVAC systems on the heat flow of the building. Thermal resistance and capacitance parameters depend

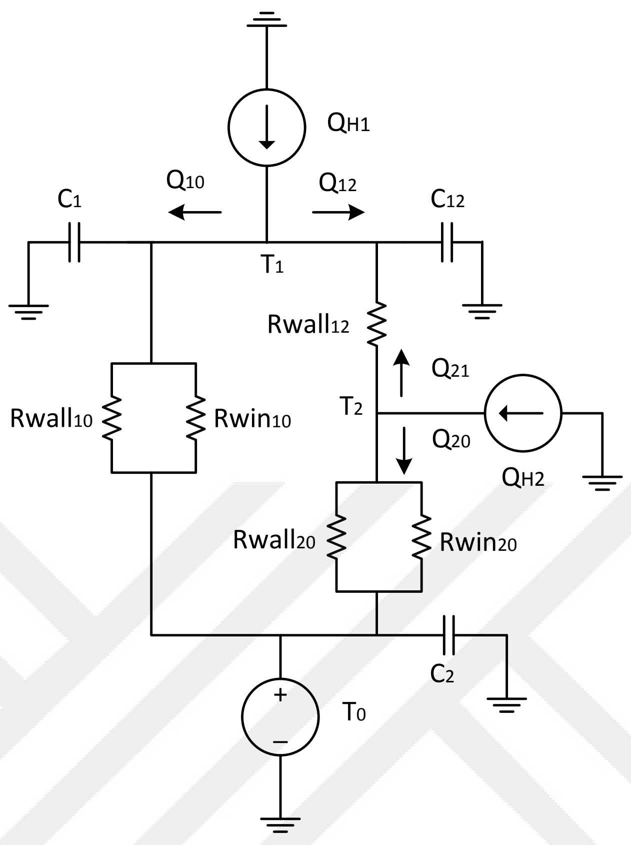


Figure 4.2: Thermal RC network of the building having two adjacent rooms.

on the physical properties of the materials used in the building. The thermal resistance parameter of a material is the function of thickness, surface area, and thermal conductivity of the corresponding material and the thermal capacitance parameter of a material is the function of mass and heat capacity of the corresponding material.

Figure 4.2 represents a resistor-capacitor (RC) network of the sample 2-room building for illustration. Wall and glass thermal properties are considered to calculate equivalent thermal resistance. Rwall_{10} and Rwin_{10} refer to the thermal resistances for the wall and window areas, which front outside, in room_1. Rwall_{20} and Rwin_{20} refer to the thermal resistances for the wall and window areas, which front outside, in room_2. Rwall_{12} refers to the thermal resistance for the wall area, which is shared by two rooms. Q_{H1} and Q_{H2} refer to heat flows provided by HVAC_1 and HVAC_2 systems, respectively.

4.5 Model-based Design of the CPS Example

Model-based design (MBD) allows the designer to design, analyze, verify, and validate CPSs through several consequent design phases. We adopt a model-based CPS design approach to design a multi-room smart building incorporating temperature sensors, controllers, and HVAC systems. From CPS standpoint, the building, phenomenon of interest, and HVAC systems represent the physical system. The controllers represent the cyber system. The sensors represent the interface between the cyber and physical systems. The following subsections describe the constituent models for the CPS example and their functionalities.

4.5.1 A MATLAB/Simulink Simulation Framework

We implemented our MBD approach for the design of a multi-room smart building in MATLAB/Simulink environment. MATLAB is a high-level language and interactive environment for technical computing [153]. Simulink is a tool used to model and simulate the behavior of dynamic systems (those governed by differential equations) [154]. Simulink is integrated with MATLAB, enabling the designer to incorporate MATLAB algorithms into Simulink models and export simulation results to MATLAB for further analysis and visualization.

Figure 4.3 shows a snapshot of our system model developed in Simulink. The building subsystem incorporates a thermal model of the building and models of sensor and controller components. HVAC systems and thermal properties of the building are modeled using the aforementioned thermodynamic equations.

Figure 4.4 shows the models for heating and cooling functionalities for an HVAC system. As shown in Figure 4.4a, the heating system of a room is turned on/off via control command provided by the controller deployed in the corresponding room. The gain, C_{air} (i.e., thermal

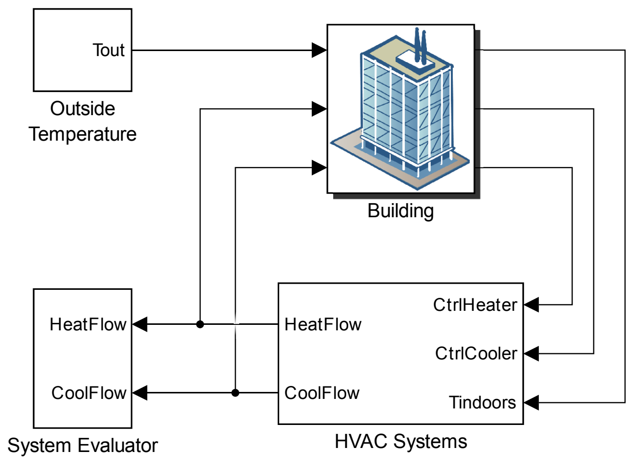


Figure 4.3: Simulink model of the multi-room building.

capacitance of air), is calculated through a fixed air mass flow rate (kg per hour) for the heating system and thermal capacity (J/kg℃) of the air provided by the heating system. T_r and T_h parameters represent the temperatures of the room and hot air provided by the heating system of the room, respectively. Q_{Heater} represents the heat flow supplied by the heating system.

As shown in Figure 4.4b, similar parameters are specified for the cooling system as well. T_c is the temperature of cool air provided by the cooling system of the room and the gain is calculated through a fixed air mass flow rate (kg per hour) for the cooling system and thermal capacity of the air (J/kg℃) provided by the cooling system. Q_{Cooler} represents the heat flow supplied by the cooling system.

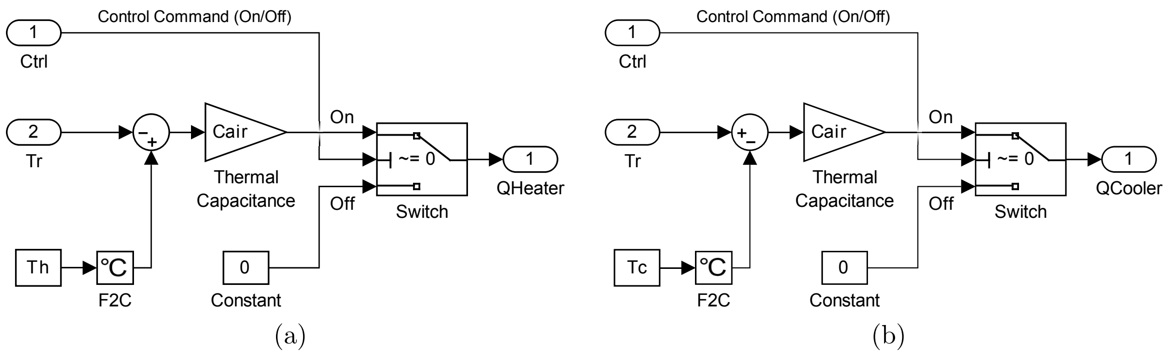


Figure 4.4: HVAC system model: a) Heater model, b) Cooler model.

$$Q = \Delta T \; C_{\text{air}} \tag{4.6}$$

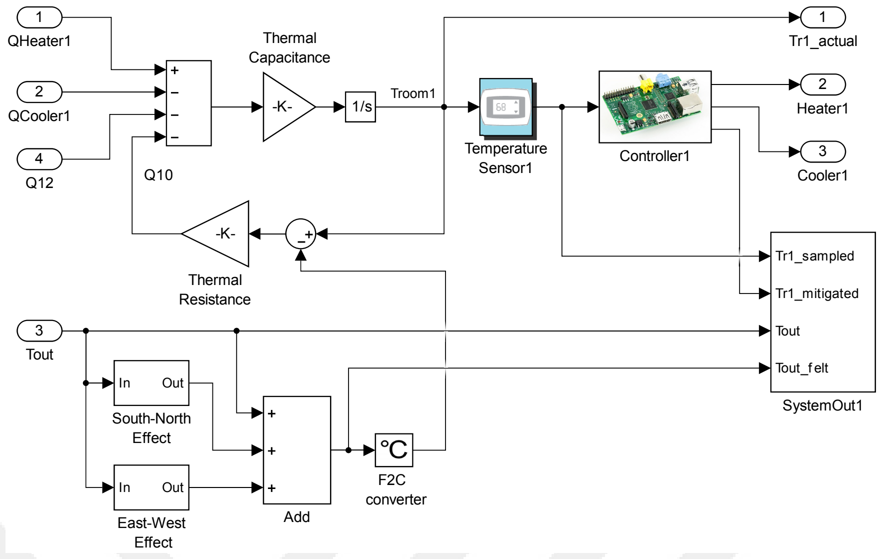
$$Q_{\text{Heater}} = (T_{\text{h}} - T_{\text{r}}) \; C_{\text{air}} \tag{4.7}$$

$$Q_{\text{Cooler}} = (T_{\text{r}} - T_{\text{c}}) \; C_{\text{air}} \tag{4.8}$$

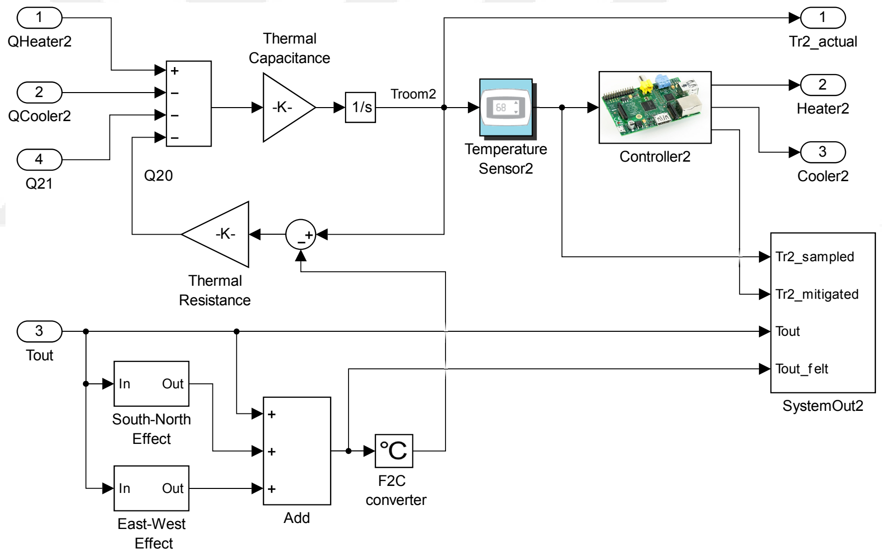
The switch in the heater and cooler models yields its upper port if the control input is 1 while it yields its lower port if the control input is zero. The heater and cooler models are based on Equations 4.6, 4.7, and 4.8 where C_{air} denotes the function of air flow rate by mass (kg per hour) and heat capacity of air.

Figure 4.5 shows Simulink models for room$_1$ and room$_2$, incorporating thermal parameters, temperature sensors, and controllers for each individual room. As shown in Figure 4.5a, $\text{Q}_{\text{Heater1}}$ and $\text{Q}_{\text{Cooler1}}$ represent the heat flows supplied by the heating system and the cooling system in room$_1$, respectively. Q_{12} represents the heat transfer from room$_1$ to room$_2$ and Q_{10} represents the heat transfer from room$_1$ to outside. Since supplied hot air increases the room temperature, $\text{Q}_{\text{Heater1}}$ is added to aggregate heat flow and since supplied cool air decreases the room temperature, $\text{Q}_{\text{Cooler1}}$ is subtracted from aggregate heat flow. Similar parameters are defined for room$_2$ as well, following the same notation as shown in Figure 4.5b.

In Figure 4.5, Tr1_actual and Tr2_actual refer to the actual temperatures of room$_1$ and room$_2$, respectively. Tr1_sampled and Tr2_sampled refer to the sensor readings regarding the temperature of room$_1$ and room$_2$, respectively. Tr1_actual and Tr2_actual temperatures are read by sensor$_1$ and sensor$_2$, respectively and provided to the controllers for further process. The controllers process the sampled temperatures and apply mitigation techniques if any fault is detected. The temperature values after mitigation technique being applied by the controllers are indicated as Tr1_mitigated and Tr2_mitigated for room$_1$ and room$_2$, respectively. The controllers take necessary actions depending on the control instructions for the HVAC system by considering those mitigated temperatures.



(a)



(b)

Figure 4.5: Simulink model of: a) Room$_1$, b) Room$_2$.

Figure 4.6 shows how heat transfer between room$_1$ and room$_2$ (i.e., Q_{12} or Q_{21}) is calculated. The heat transfer is modeled considering Equation 4.5. As shown in Figure 4.5a for room$_1$, Q_{12} and Q_{10} may have positive or negative values. If Q_{12} has a positive value, then this indicates that heat flow is from room$_1$ to room$_2$ and if, on the contrary, Q_{12} has a negative value, then this indicates that heat flow is from room$_2$ to room$_1$. The same things work for Q_{10}. If Q_{10} has a positive value, then this indicates that heat flow is from room$_1$ to outside

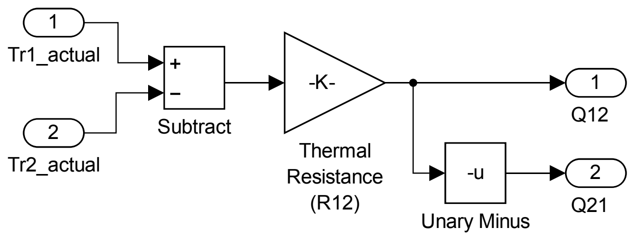


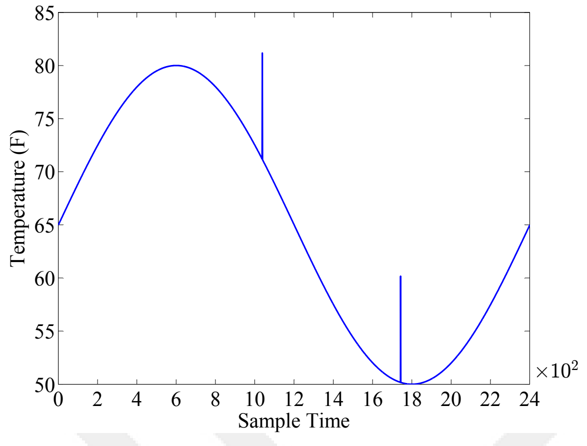
Figure 4.6: Heat transfer between two adjacent rooms.

and if, on the contrary, Q_{10} has a negative value, then this indicates that heat flow is from outside to $room_1$. Same approach is used to aggregate heat flows for $room_2$ as shown in Figure 4.5b.

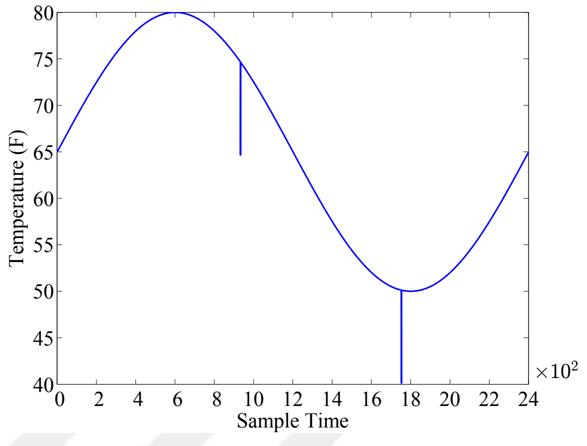
In Figure 4.5, thermal resistance refers to R_{10} for $room_1$ (i.e., equivalent resistance of $Rwall_{10}$ and $Rwin_{10}$) and R_{20} for $room_2$ (i.e., equivalent resistance of $Rwall_{20}$ and $Rwin_{20}$), respectively. In both models, the effect of rooms' orientation on the room temperatures are specified by east-west and south-north orientation parameters.

The fault mitigation functionality is provided by the controller in the room. Therefore, Tr1_mitigated and Tr2_mitigated are used for decision making to trigger the corresponding HVAC systems. Outputs of the controllers carry control commands (i.e., on/off commands) for HVAC systems' heating and cooling facilities.

To have a well-defined thermal model of the building, the effect of direct solar radiation on the building orientation is considered for North, South, East, and West directions and the outcome is referred as Tout_felt. The sun's angle of incidence and the orientation of the room (South/North and East/West) are specified and the result is added to Tout to calculate Tout_felt, which is then used to calculate Q_{10} and Q_{20}, as shown in Figure 4.5. The following subsections discuss the fault and system evaluation models.

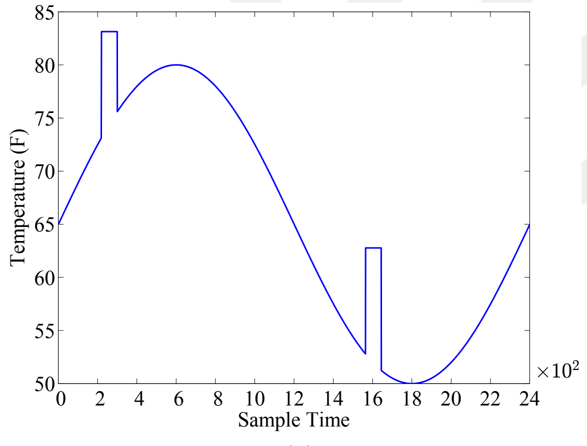


(a)

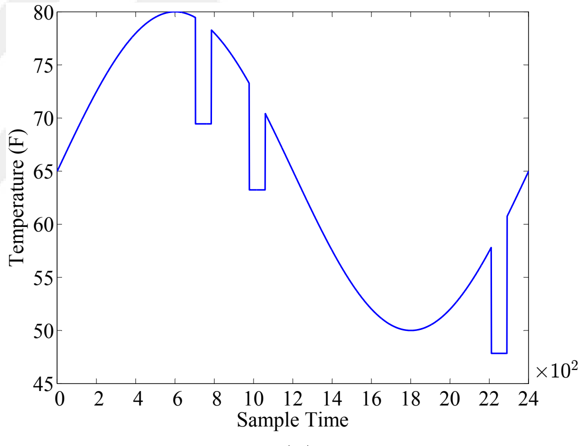


(b)

Figure 4.7: Single-sample-spike fault: a) Positive spike, b) Negative spike.



(a)



(b)

Figure 4.8: Sample-and-stay fault: a) Positive spike, b) Negative spike.

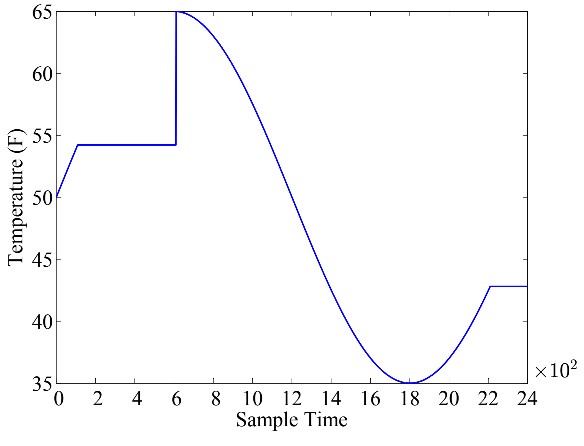


Figure 4.9: Stuck-at fault.

4.5.2 Fault Model

The implementation of fault semantics in Simulink with a model-based design approach allows us to inject the faults into our system model and analyze their impacts on the overall system behavior. As we mentioned in Section 4.3, we examine some sensor faults that are observed in the real-world sensor deployments [161, 162]. We provided our terminology and fault semantics for those sensor faults of interest in Section 4.3. In this section, we illustrate the aforementioned faults to make their semantics more clear.

Figure 4.7 gives an illustration of single-sample-spike fault. The sinusoidal signal is chosen as an example for the illustration to roughly represent outdoor temperature change. The x-axis represents the number of samples taken throughout a day. The y-axis represents temperature in Fahrenheit. Figures 4.7a and 4.7b show the effects of single-sample positive and negative spikes on the input signal, respectively when the original sinusoidal signal is given to the fault generator as an input.

Figure 4.8 gives an illustration of spike-and-stay fault. Figures 4.8a and 4.8b show the effects of positive and negative spikes on the input signal, respectively when the original sinusoidal signal is given to the fault generator as an input. Figure 4.9 gives an illustration of stuck-at fault and shows the effect of stuck-at fault on the input signal when the original sinusoidal signal is given to the fault generator as an input.

We inject these faults into our CPS model and evaluate the system success under faulty sensor conditions without and with mitigation techniques being applied by the controller. The following section explains our evaluation criteria for the system outcome.

4.5.3 System Evaluation Model

The system outcome is evaluated based on mitigate success metric that is the function of thermal comfort and energy efficiency. In order to evaluate thermal comfort for the occupants of a room, first we define a comfort interval which is in 10°F range. The upper and lower bounds of the range is 5°F more and less than the set point (i.e., reference) for the room temperature, respectively. The comfort range indicates, we assume, that the occupants of a room feel comfortable in that temperature interval. After defining comfort range, we answer the following question considering sensor fault occurrences:

- How many time steps are there at which the actual room temperature was out of comfort range with the tolerance of 0.5°F?

Thermal comfort metric is the function of actual room temperature for the corresponding room. We assume that thermal comfort is 100% without any fault occurrences. Under fault occurrences, the answer to the above-mentioned question multiplied by 0.1% gives us the total deduction from thermal comfort percentage. In other words, a deduction from the thermal comfort metric is calculated according to how long the actual room temperature stays out of the comfort range.

$$Mitigate\ Success = \frac{Base\ Energy - Variation\ in\ Energy + Thermal\ Comfort}{2} \tag{4.9}$$

Energy consumption is evaluated by comparing energy consumption of the building under fault occurrences with energy consumption of the building without fault occurrences that is, what we call, base energy consumption. Equation 4.9 explains how we formulate mitigate success metric. Base energy metric is assumed 100% without any fault occurrences. Variation in energy is defined as a difference in percentage between base energy consumption and energy consumption under fault occurrences. Mitigate success metric is used to evaluate the system success for temperature control in our multi-room building. The following section explains the proposed mitigation techniques for the aforementioned sensor fault scenarios.

4.6 Fault Mitigation Techniques

Cyber-physical systems are more than likely to change how we interact with the real world. Therefore, they are to meet our high expectations. Considering the design of smart buildings with CPS approach, the CPS designer should be crystal clear about the following question:

- Is it possible to ensure satisfactory comfort for occupants and energy efficiency under sensor fault occurrences?

Since our main concern in the multi-room building design is to alleviate adverse impacts of sensor faults on occupants' perceptions of a smart building, the temporal and spatial correlations between sensors' readings can be leveraged to design fault-tolerant building automation systems. The following subsections explain the proposed techniques to mitigate the sensor faults of interest in this chapter.

4.6.1 Read-back Technique

The read-back technique (RBT) considers the temporal correlation between consecutive points of values extracted by an individual sensor. The rate of change of the physical phenomenon (i.e., temperature) is observed to identify and mitigate the sensor faults. The rate of change refers to the difference in temperature between two consecutive samples. Since temperature, as a phenomenon, changes slowly, the rate of change for temperature sensor readings is expected to be low.

In the read-back technique, the controller computes the rate of change between two consecutive points from sensor readings and compares it to a threshold value. The threshold value is calculated through the daily variation pattern of the physical phenomenon. If the rate of change is above the threshold, then the sampled data is considered to be spike and the previously sampled data is used by the controller to take the control action for HVAC system facilities.

4.6.2 Nearest Neighbor Monitoring Technique

The nearest neighbor monitoring technique (NNMT) considers both the temporal and spatial correlations between the values extracted by sensors deployed in two neighbor rooms. Each room has a dedicated sensor, controller, and HVAC facilities. The controller samples the temperature values from the sensor in the corresponding room. In nearest neighbor monitoring, the rooms are considered to be the nearest neighbors if they share at least a wall. As in RBT, the rate of change of temperature is observed to identify and mitigate the sensor faults. If the rate of change is below the threshold and not equal to zero then the controller continues using samples from the sensor. However, if it is equal to zero, then that means the sensor is stuck and the nearest neighbor monitoring technique is applied to mitigate the fault.

In case of applying NNMT, the controller stops using samples from the faulty sensor till the sensor returns to the expected behavior (i.e., correct behavior of the sensor). Meanwhile the controller samples the temperature data from the sensor in the nearest neighbor room. When the faulty sensor returns to correct behavior, the controller stops sampling from the nearest neighbor and uses the data extracted by the sensor in the corresponding room. The reason why the temperature data is obtained from the neighbor sensor readings is the spatial correlations between the temperatures in the neighboring rooms. A many core embedded

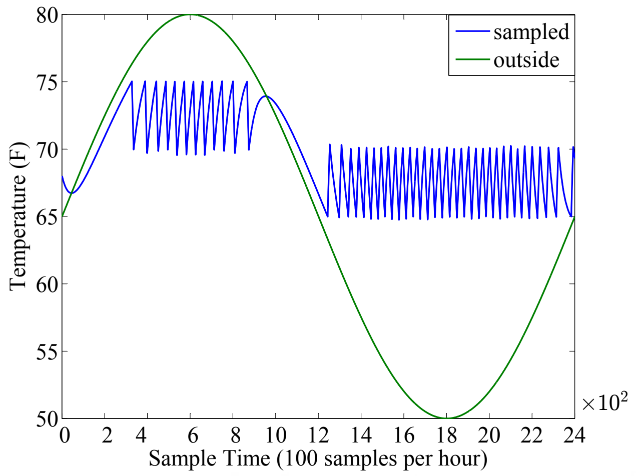


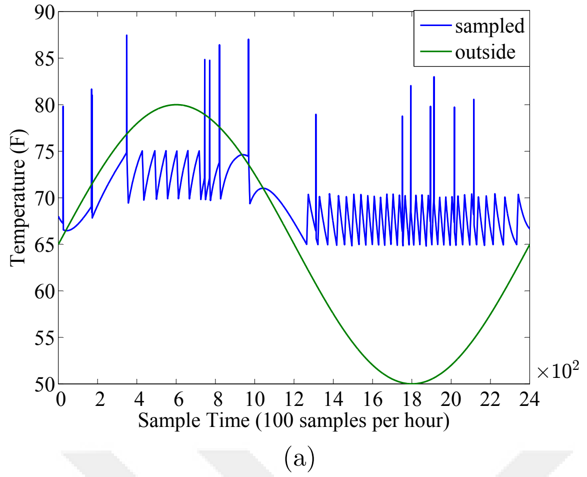
Figure 4.10: The variation of a room's temperature considering the comfort range.

processor as introduced in Chapter 5 can be used as the control system of the building where each core in the processor represents a controller deployed in a room. The following section discusses our simulation results and system success evaluation before and after mitigation techniques being applied.

4.7 Experiments And Evaluation

We implemented the proposed fault mitigation techniques in Simulink and applied them to our demonstrative CPS model. As previously mentioned, we define a comfort interval including an upper bound, lower bound, and set point for the room temperature. Later, the upper bound, lower bound, and set point are used to control the cooling and heating facilities of dedicated HVAC system in the corresponding room.

Figure 4.10 shows how the temperature of a room changes according to the comfort range specification under perfect conditions (i.e., without any fault occurrences). In the figure, the green and blue lines represent the variation of outside temperature and sampled sensor data regarding room temperature, respectively. The set point for the room temperature is determined as 70°F. The upper and lower bounds are determined to specify the comfort



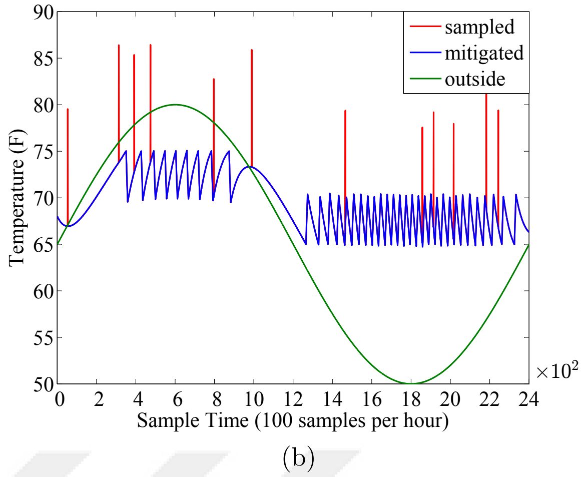
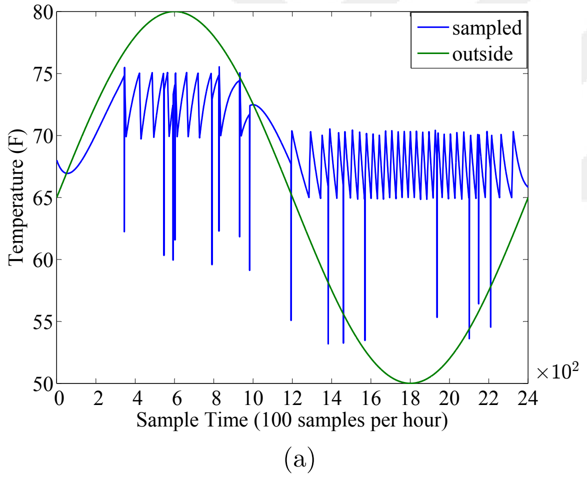


Figure 4.11: Single-sample-spike fault occurrence as positive spike: a) Sampled sensor data, b) Corrected sensor data by RBT.



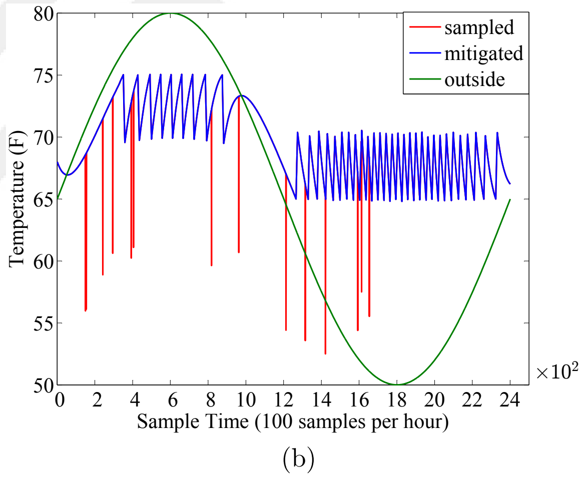


Figure 4.12: Single-sample-spike fault occurrence as negative spike: a) Sampled sensor data, b) Corrected sensor data by RBT.

range. The upper bound is 75°F (5°F more than the set point) and the lower bound is 65°F (5°F less than the set point). The upper and lower bounds are used to designate an operation region for the cooling and heating facilities of the HVAC system, respectively. To be more precise, the cooling system operates between the upper bound and set point and the heating system operates between the set point and lower bound.

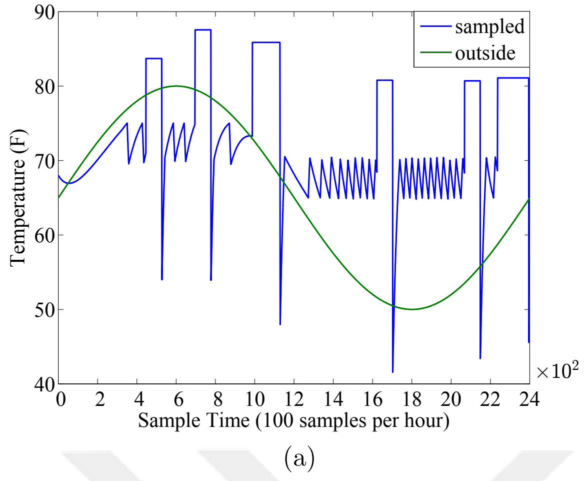
Since we treat the cooling and heating facilities of an HVAC system as identical heat resources, specifying non-overlapping operation regions averts the deadlock. The upper and lower bounds are used by the controller to switch on the cooling and heating systems, respectively. The set point is used by the controller to switch off both the heating and cooling systems.

Figures 4.11 and 4.12 show the temperature pattern of a room throughout a day, considering the occurrence of single-sample-spike (SSS) faults. In the figures, the x-axis represents the number of samples taken by the controller throughout a day. The controller samples the sensor data as 100 samples per hour. The y-axis represents the temperature in Fahrenheit.

Figures 4.11a and 4.12a show sampled sensor readings under single-sample-spike with positive and negative spikes, respectively. As previously mentioned, single-sample-spike fault occurs randomly in a single sample with an unexpected spike. In Figure 4.11a, the sensor reading rises up to 88°F and in Figure 4.12a, the sensor reading goes down to 53°F. Figures 4.11b and 4.12b show the temperature variation when the read-back technique being applied, referred as mitigated. In the figure, the green, red, and blue lines represent the variation of outside temperature, sampled sensor reading, and mitigated sensor reading, respectively. As the results indicate, this technique works well if the fault does not take more than one samples.

Figures 4.13 and 4.14 show the variation of room temperature under spike-and-stay (SAS) fault occurrences with positive and negative spikes, respectively. In the figures, the x-axis represents the number of samples taken by the controller throughout a day. The controller samples the sensor data as 100 samples per hour. The y-axis represents the temperature in Fahrenheit.

Figures 4.13a and 4.14a show sampled sensor readings under spike-and-stay fault with positive and negative spikes, respectively. The fault starts with a spike and stays constant for a number of samples. An example of spike-and-stay fault with positive spike and its effect on the room temperature are shown in Figure 4.13a. Since the controller relies on the faulty sensor reading, it initiates the cooling system if the sudden increase in the sampled sensor



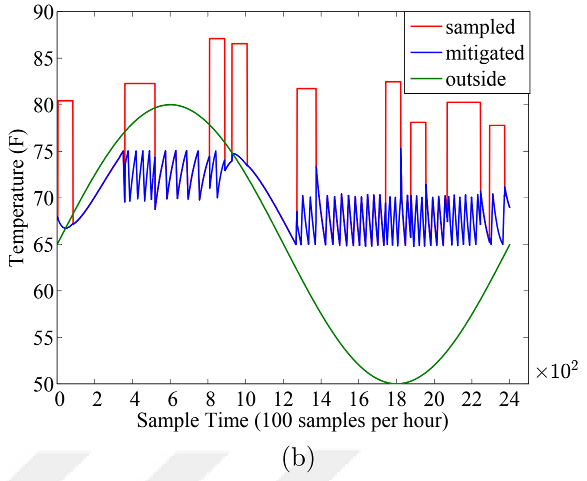
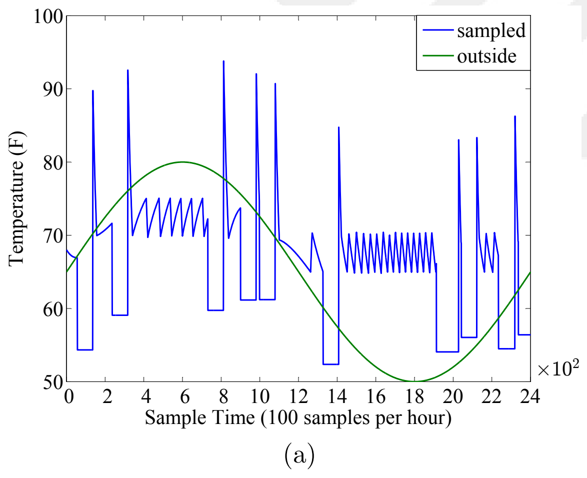


Figure 4.13: Spike-and-stay (SAS) fault occurrence as positive spike: a) An example of SAS fault and its effect on the room temperature, b) Another example of SAS fault and correcting the fault by NNMT.



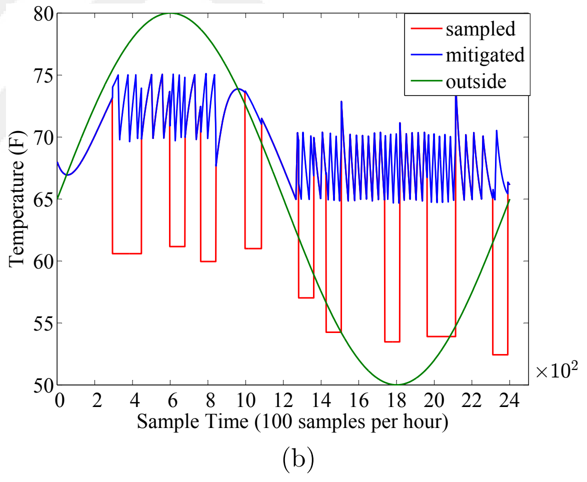


Figure 4.14: Spike-and-stay (SAS) fault occurrence as negative spike: a) An example of SAS fault and its effect on the room temperature, b) Another example of SAS fault and correcting the fault by NNMT.

reading is above the upper bound. The cooling system will operate continuously till the sensor returns to the correct behavior. As shown in Figure 4.13a, when the sensor returns to the correct behavior, low room temperature due to over-operation of the cooling system during faulty sensor readings is observed by the sensor. Then, the controller initiates the heater since the room temperature is below the lower bound. Same works for the consecutive occurrences of the fault till the end of the day.

Another example of spike-and-stay fault with negative spike and its effect on the room temperature are shown in Figure 4.14a. Since the controller relies on the faulty sensor reading, it initiates the heating system if the sudden decrease in the sampled sensor reading is below the lower bound. The heating system will operate continuously till the sensor returns to the correct behavior. As shown in Figure 4.14a, when the sensor returns to the correct behavior, high room temperature due to over-operation of the heating system during faulty sensor readings is observed by the sensor. Then, the controller initiates the cooler since the room temperature is above the upper bound. Same works for the consecutive occurrences of the fault till the end of the day. Unless spike-and-stay fault is mitigated, this situation leads to inefficient use of HVAC facilities, increased energy consumption, and occupants' dissatisfaction regarding thermal comfort.

Figure 4.13b shows the variation of room temperature under spike-and-stay fault with positive spike when the nearest neighbor monitoring technique is applied. As seen in the figure, even though there is a slight difference in the transitions between sensor readings, the room temperature approximately follows the expected pattern. Obviously, the occurrence of the fault at the time step of around 1800 and its mitigation have different effect than the other ones. That's because this fault occurs while outside temperature is at the minimum (i.e., 50°F) in its daily variation and the nearest neighbor room has more walls shared with outside than the room being observed.

The heating facility of the corresponding room is being controlled according to the data coming from the sensor deployed in the nearest neighbor room. So the heater operates more than necessary. When the sensor returns to the correct behavior, the controller realizes that the temperature is slightly beyond the upper bound and the cooler facility is switched on. Considering the temperature variation in the range of 20 to 25°F under no fault mitigation, at most 1°F temperature difference out of comfort range in one instance of the fault is favorable.

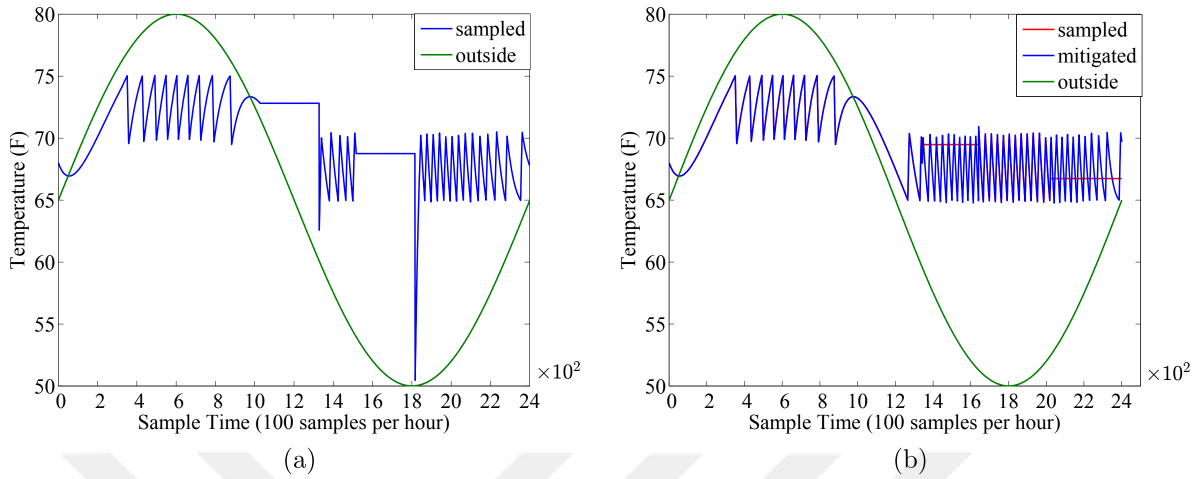


Figure 4.15: Stuck-at (SA) fault occurrence: a) An example of SA fault and its effect on the room temperature, b) Another example of SA fault and correcting the fault by NNMT.

Figure 4.14b shows the variation of room temperature under spike-and-stay fault with negative spike when the nearest neighbor monitoring technique is applied. Likewise positive spike, even tough there is a slight difference in the transitions between sensor readings, the room temperature approximately follows the expected pattern. Obviously, the occurrence of the fault at the time step of around 2000 and its mitigation have different effect than the other ones. The reason is same as we explained in the previous paragraph. This fault occurs while outside temperature is close to the minimum (i.e., 50°F) in its daily variation and the nearest neighbor room has more walls shared with outside than the room being observed. This causes the heater to operate slightly more than needed. Again, considering the temperature variation in the range of 20 to 25°F under no fault mitigation, a few Fahrenheit temperature difference inside comfort range is favorable.

Figure 4.15 shows the variation of room temperature under stuck-at (SA) fault occurrences. In the figures, the x-axis represents the number of samples taken by the controller throughout a day. The controller samples the sensor data as 100 samples per hour. The y-axis represents the temperature in Fahrenheit.

Figure 4.15a shows sampled sensor readings as an example of stuck-at fault and its effect on the room temperature. The second instance of the fault occurs when outside temperature goes down below the lower bound. However, since the controller relies on the faulty sensor reading, it does not initiates the heater. So the room temperature follows the outside temperature and gets very close to the minimum outside temperature (i.e., 50°F) at the end of about 3 hours. When the sensor returns to the correct behavior, low room temperature due to under-operation of the heating system is observed by the sensor. Then, the controller initiates the heater since the room temperature is below the lower bound.

Figure 4.15b shows the variation of room temperature under stuck-at fault when the nearest neighbor monitoring technique is applied. As seen in the figure, even tough there is a slight difference in the transitions between sensor readings, the room temperature approximately follows the expected pattern.

Besides the results regarding room temperature variations, we also provide variation in energy consumption and thermal comfort under perfect, faulty, and mitigated sensor conditions. Appendix A contains the simulation results providing energy consumption (kWh), variation in energy consumption (%), cost ($), and thermal comfort (%) for the individual rooms and all building. As of January 2015, the price of electricity for residential usage in California is about 17.5 cents per kWh [164] and we calculate the cost of energy consumption based on this data from [164]. Mitigate success and thermal comfort metrics are assessed based on the system evaluation metrics mentioned in Section 4.5.3.

Table A.1 shows base energy consumptions, costs, and thermal comfort metric under perfect conditions. The reason why the heater consumes more energy than the cooler is related to variation in outside temperature throughout a day. As shown in Tables A.2 and A.4, single-sample-spike (SSS) faults do not degrade thermal comfort because the duration of faults are only about half minute (i.e., sampling time) but cause an increase up to about 2% in total energy consumption (i.e., energy consumption of the building). Tables A.3 and A.5 indicate that the read-back technique is successful enough in eliminating the increase in the total energy consumption.

Tables A.6 and A.8 show the results for spike-and-stay (SAS) faults with positive and negative spikes, respectively. Unlike SSS faults, SAS faults cause significant increase in total energy consumption (i.e., up to 41% increase) and intolerable decrease in thermal comfort (i.e., up to 83% decrease) for the room in which the faulty sensor is deployed. The increase in energy consumption is due to over-operation of HVAC systems. As shown in Tables A.7 and A.9, the nearest neighbor monitoring technique performs very well regarding total energy consumption with up to 39% improvement, thermal comfort of the room with up to 79% improvement, and sensor fault mitigation with the success rate of about 98% in average.

Tables A.10 and A.11 show the results for stuck-at fault without and with the nearest neighbor monitoring technique being applied. The fault causes an about 5% decrease in total energy consumption compared to base energy consumption but a significant decrease in the thermal comfort of the room (i.e., 35% decrease) is being observed. The decrease in energy consumption is due to under-operation of HVAC systems. Even though the nearest neighbor monitoring technique leads to about 3% increase in total energy consumption, it significantly improves the thermal comfort as being 100%. The results show that the proposed fault mitigation techniques provide high mitigation success percentage that leads to insignificant increase in energy consumption compared to base energy consumption and significant increase in thermal comfort under faulty sensor conditions.

4.8 Conclusions

Building automation, also known as the smart building, plays an important role in everyday lives of people as a CPS application. Satisfying thermal comfort of occupants and ensuring energy efficiency are important issues in building automation. Occupants consider the environment comfortable if no type of thermal discomfort is present. Since the controllers in the buildings heavily rely on the knowledge extracted by temperature sensors to trigger HVAC systems and satisfy thermal comfort, any occurrence of sensor faults may lead to inefficient use of HVAC systems, increasing energy consumption, and thermal discomfort.

In this chapter, we examined a multi-room building as a CPS application. Given the need for thorough understanding of sensor faults' semantics and their effects on the thermal comfort of occupants and energy consumption, we investigated temperature sensor faults observed in the real world deployments and provided system evaluation metrics to quantify the impacts of those faults on the system outcome in terms of energy consumption and thermal comfort. We adopted a model-based design methodology for the CPS application and developed reusable system models in the MATLAB/Simulink environment. Model-based approach enabled us to easily integrate our fault and system evaluation models into the traditional CPS model.

We conclude that aforementioned faults may significantly increase energy consumption and significantly decrease thermal comfort of the occupants unless appropriate measures are taken. In order to bridge the gap between occupants' expectations and their perceptions of a multi-room smart building, we proposed fault mitigation techniques that are based on temporal and spatial correlations between sensors' readings. We conclude that the proposed approaches improved thermal comfort by up to 79% for the room where the faulty sensor was deployed and reduced total energy consumption by up to 39% and yielded to an average of 98% mitigation success that significantly improves energy efficiency and thermal comfort of the occupants under faulty sensor conditions.

Chapter 5

XGRID Many-core Processor

5.1 Introduction

Embedded systems have an important place in our daily lives. Compute demands of embedded systems have increased in recent years for many electronic devices, including but not limited to mobile devices, consumer appliances, network devices, and military applications. With the growing popularity of the cyber-physical systems (CPSs), this increase in compute demands is expected to continue for specialized embedded systems to support a wide range of new applications.

To satisfy increasing demands for compute cycles, operating clock frequency of processors was increased accordingly for years. However, it is no longer feasible to continue gaining improvements through single-core devices by increasing clock speed [165]. Providing higher CPU clock speeds to improve performance results in a non-linear increase in power consumption and this increase generates excessive operating temperatures. This situation requires more advanced cooling systems, adding cost, and decreases the reliability of the overall system [166].

The limitations of single-core processors running at high clock rates have forced the computer industry to shift to a new approach for increasing performance of computer systems, namely, a move toward multi-core processing [165]. Multi-core processors make improvements in the performance by increasing the number of the processor cores on a single chip, with each core operating at an ideal clock speeds in order to meet overall power and thermal constraints [167].

We draw a distinction between a multi-core system, one that is limited to 8 or less cores, and a many-core system that can scale to tens, hundreds, and even thousands of cores on a single chip. A key requirement for any many-core architecture is its ability to scale efficiently. With increases in the number of cores on a single chip, the performance of the overall system becomes limited by shared resources such as buses and the memory subsystem [168]. In particular, cache coherency issues come into play as the number of cores increases beyond small number [169].

In this chapter, we present a scalable many-core processor, intended for embedded and cyber-physical applications. Our many-core embedded processor is named XGRID. Further, we outline a mapping strategy to efficiently map applications to XGRID. This chapter provides a discussion of a scalable many-core embedded processor adopting 2D grid network, inspired by a novel FPGA-like interconnect network, an optimal mapping of benchmark applications onto target XGRID architecture, and a deployment scenario of XGRID on a multi-room building as a CPS application. XGRID helps adopt redundancy techniques (e.g., controller redundancy) for fault tolerance in CPS design due to its inherent support for parallelism.

We also describe a comprehensive simulation environment for XGRID. Our simulation platform, in addition to offering a cycle accurate functional execution environment, provides detailed energy and performance results to better guide the application mapping process. The rest of this chapter is organized as follows. Related work and motivation are discussed in Section 5.2. The XGRID architecture is introduced in Section 5.3. The XGRID simulation framework is introduced in Section 5.4. Application mapping is outlined in Section 5.5. System energy profiling is introduced in Section 5.6. A deployment scenario for XGRID in a multi-room building is discussed in Section 5.7. Performance analysis and experimental results are provided in Section 5.8. Conclusions are given in Section 5.9. A portion of this chapter was published in [4].

5.2 Related Work and Motivation

Modern system-on-chip (SoC) design methods are increasingly becoming communication-centric rather than computation centric, so it is expected that the interconnect effect will dominate the performance of many-core SoCs [170]. There are two common interconnect architectures, namely bus based and network-on-chip (NoC) based interconnects. Various authors have investigated architectural issues related to bus based and NoC based approaches. We summarize a number of related articles that address the key components of the XGRID project.

A hierarchy-bus based multi-processor system-on-chip (MPSoC) is studied in [171] by prototyping the architecture on a field-programmable gate array (FPGA) and comparing the results of the MPSoC with a single core processor. A framework for NoC-based MPSoC, called HeMPS is presented in [172] with support for static and dynamic task mapping based on a C/SystemC simulation model of processors and memories.

The NoCRay graphic accelerator, an implementation of NoC-based homogeneous MPSoC, is studied in [173] by using a complete design methodology that tackles different aspects of hardware architecture, system level modeling, and programming models. Here, the results are compared with a general purpose single core high-speed processor. The Medea framework, an example of NoC based multiprocessor architecture adopting a configurable hybrid shared-memory/message-passing architecture, is presented in [174] with a fast, cycle-accurate SystemC implementation enabling system exploration by varying several parameters such as the number and types of cores, cache size and policy, and NoC features.

The study in [175] represents a heterogeneous multi-core simulator framework, called MC-Sim, which is capable of accurately simulating a variety of processors, memory, NoC configurations, and application specific coprocessors. Gem5 simulation infrastructure, the combination of the M5[176] and Gems[177] simulators, are proposed in [178] as a full system simulation tool, offering a diverse set of CPU models, system execution models, and memory models. HORNET, a configurable, cycle-accurate multi-core simulator is presented in [179]. Being based on an NoC architecture, it also supports a variety of memory hierarchy and interconnect configurations. Graphite, a distributed, parallel many-core simulator is introduced in [180] for design space exploration of future many-core processors. Being based on the TILE[181] processor architecture with a mesh network, it provides detailed performance metrics of application benchmarks and supports McPAT[182], an integrated power, area, and performance measuring framework.

As outlined in earlier studies, bus based architectures are not scalable with increasing number of cores [183]. Hence, NoC structures are proposed as a solution to the limitations of bus based architectures [184]. NoC structures offer some advantages as well as some challenges. For example, switch units, network interfaces, and inter-switch wires result in substantial silicon area overhead. Increased networking complexity as well as the number of interconnected cores within an NoC introduce a considerable trade-off between area and performance [185, 186].

In an NoC, often times, the interconnect infrastructure consumes more power than other system components [187]. Power consumption can be reduced by decreasing the number of interconnected components; however, this adversely affects overall performance [187]. Scaling down the supply voltage to save power consumption can not sufficiently compensate for higher interconnect infrastructure complexity [188] and need for compute cycles dedicated to the message routing algorithms. Furthermore, in NoCs, non-deterministic communication delays do occur due to communication resource contention [189]. So quality of service (QoS) and congestion control mechanisms are needed to overcome that issue [189].

Due to the limitations of bus interconnect and challenges of NoC interconnect, we propose a novel FPGA-like many-core architecture which combines positive attributes of bus based (i.e., power efficient and deterministic) and NoC based interconnects (i.e., scalable and flexible). The simple and low-clock speed nature of XGRID makes it inherently power efficient. While FPGAs do suffer increased power consumption as a result of large scale switching networks [190, 191], our proposed XGRID architecture has a low-cost static interconnect network, making it compile-time configurable with minimal power overhead.

5.3 The XGRID Architecture

Our embedded many-core processor platform integrates processing cores, on-chip memory per core, FPGA-like interconnection network and serial high-speed I/O units. We call our architecture XGRID, as it consists of a two dimensional grid of homogeneous cores. Each XGRID core strictly follows a reduced instruction set computer (RISC) architecture. Specifically, cores follow a standard five-stage instruction pipeline (fetch, decode, execute, memory-access, and write-back) and lack the branch prediction and out-of-order execution capabilities. Per core flash memory is used to store program instructions, making XGRID in-system programmable. Each core maintains a dedicated instruction and data cache. Each core operates at a relatively low clock frequency, namely, 100 MHz to 500 MHz. As a result, the XGRID cores are lightweight and, hence, low power. The compute performance of XGRID comes from the scalability in terms of the number of cores that are utilized to execute parallel algorithms.

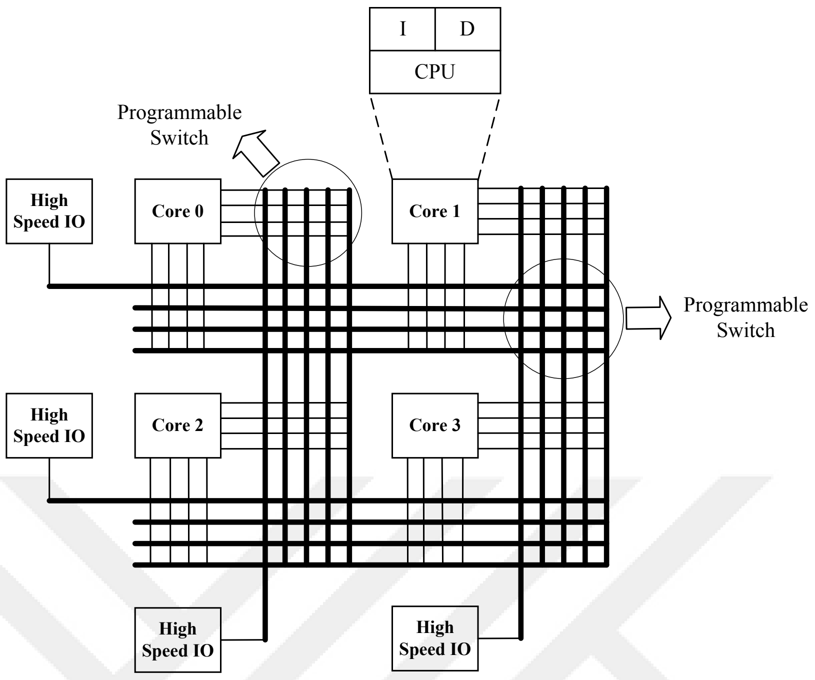


Figure 5.1: An instance of XGRID with 2x2 cores, I: instruction memory and D: data memory.

In XGRID, communication between cores is achieved via an FPGA-like interconnection network. FPGAs use rows and columns of buses with programmable switching fabrics at the intersections of the row/column buses to route input/output of logic-blocks [192]. XGRID replaces the FPGA logic-blocks with cores and otherwise adopts the FPGA interconnect fabric for all communication among the cores. As previously mentioned, the difference from an FPGA interconnect is that XGRID uses a interconnect network that is compile-time configurable.

An instance of the XGRID interconnection network with two rows and two columns is shown in Figure 5.1. The figure shows row and column buses in the interconnect network, represented as thick lines. Each core has N word-size ports to send or receive data over the buses. The port connections are represented as thin lines in Figure 5.1. Communication buses are dedicated, during the programming phase, for bidirectional I/O between two cores with deterministic transmission rates. Appropriate switches need to be set to establish a communication between a pair of cores. This programming, as with FPGA-programming, is

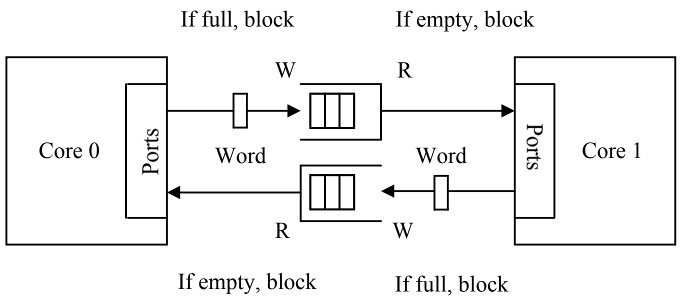


Figure 5.2: Message passing mechanism in a communication channel, W: write and R: read.

performed during the design phase, and the programming bit stream is stored in an on-chip flash memory, making the XGRID communication infrastructure in-system programmable.

XGRID uses a strict message passing system of communication among cores. The cores can communicate with each other via an inter-core communication facility. This facility provides, to the software, a primitive instruction, called XPORT, which is used to send or receive data among cores. Higher-level software routines can be built on top of this instruction to facilitate appropriate transfer capabilities, such as block transfer protocols.

We call the established path between two cores a communication channel as shown in Figure 5.2. Communication channels include bidirectional buffers to maximize instantaneous throughput among cores. The message sent by a core is of word-size. Since buffer size is limited, a sending core blocks when its send buffer is full. Likewise, a receiving core blocks when its receive buffer is empty. The blocking nature of sending (XPORT=value) or receiving (value=XPORT) are buffer synchronized, using a consumer-producer message passing scheme [193].

A major advantage of XGRID is that it avoids global caches and their associated coherency problem as well as shared memory infrastructure having complex on-chip bus and memory controllers. In this sense, XGRID is scalable, as the cost of additional rows or columns scales linearly, namely, consisting of the cost of the new cores and FPGA-like communication fabric.

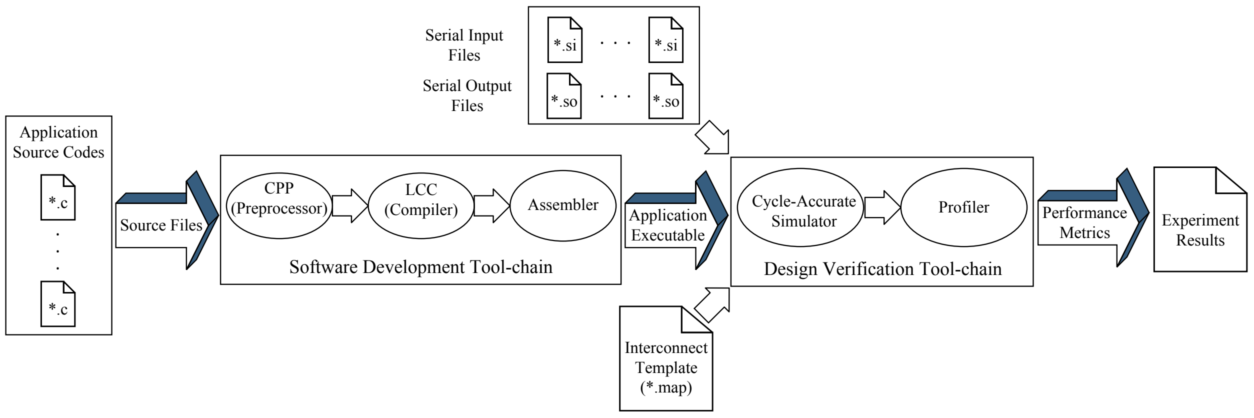


Figure 5.3: The XGRID simulation framework.

5.4 The XGRID Simulation Framework

The XGRID simulation framework simulates the complete XGRID many-core system-on-chip (SoC) device, including processing cores, on-chip memories per core, interconnection network fabric, and serial high-speed I/O units. We implemented the simulation framework in C++, considering the hardware organization of the proposed XGRID architecture.

As shown in Figure 5.3, the XGRID simulation framework consists of a complete software development tool-chain as well as a design verification tool-chain. The software flow is as follows. A sequential application is manually partitioned and revised for a multi-core target. The multi-core software application is fed through the XGRID compiler/assembler/linker tool-chain (i.e., tools based on the LCC[194]). The resulting object files are automatically mapped to cores on XGRID and the required communication channels are automatically established as needed. The mapping and routing of a parallel application to XGRID is automated using a heuristic algorithm (described later).

Once mapped to XGRID, the entire software/hardware system can be simulated to validate functional properties as well as obtain power/performance measurements. Specifically, the XGRID profiler provides per core statistics (e.g., the number of the instructions simulated, input/output rates, IO wait, compute time, computation and communication energy con-

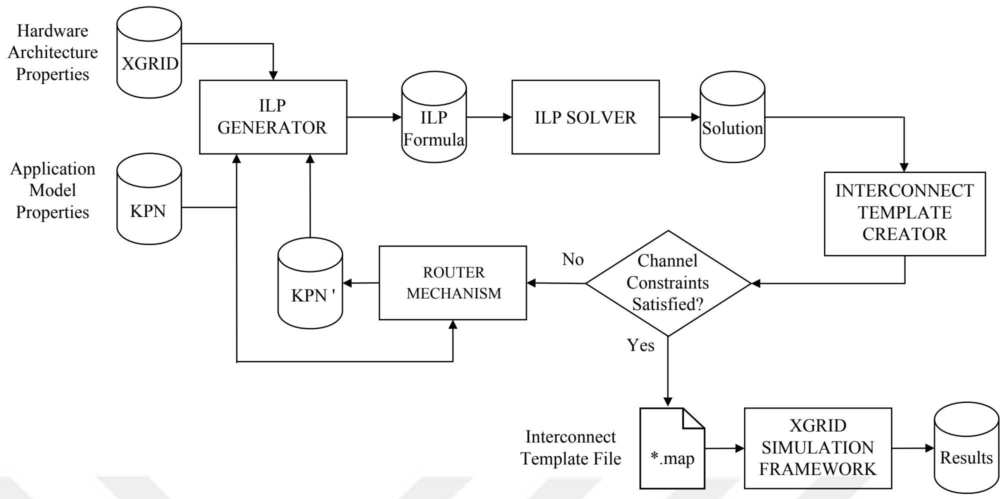

Hardware Architecture Properties
XGRID
ILP GENERATOR
ILP Formula
ILP SOLVER
Solution
Application Model Properties
KPN
INTERCONNECT TEMPLATE CREATOR
KPN '
ROUTER MECHANISM
No
Channel Constraints Satisfied?
Yes
Interconnect Template File
*.map
XGRID SIMULATION FRAMEWORK
Results

Figure 5.4: Holistic view of application mapping onto XGRID.

sumption) as well as chip-level dynamic attributes. We have developed a web-based user interface to drive the background design, simulation, and profiling tools in order to provide a fast and intuitive mechanism for students/developers to gain insight into the workings of parallel algorithms and many-core architectures.

5.5 Application Mapping

We follow a general scheme for our design flow of application mapping. This scheme includes application modeling and the actual mapping stages. Our holistic application mapping is shown in Figure 5.4. First of all, a sequential application is manually partitioned and represented as a Kahn process network (KPN) [195] where each process corresponds to a partition and each communication channel between two processes corresponds to a connection between two partitions. Then, the hardware architecture properties of XGRID and the benchmark application model represented as a KPN are fed into an ILP generator. The ILP generator produces ILP formulas consisting of variables, constraints, and constraint equations related to XGRID architecture. Then ILP solver generates a solution that reflects an optimal mapping and routing of the application to XGRID.

The interconnect template creator takes this ILP solution and evaluates it against any remaining communication channel constraints (i.e., those not directly captured by the ILP) of the XGRID. For example, since each core in XGRID has a limited number of ports, every communication channel in the KPN representation may not be mapped into the interconnect of XGRID. Therefore, the KPN representation may need a modification to fulfill remaining requirements. The interconnect template creator decides whether or not to accept the ILP solution. If the solution is not accepted, the KPN is modified and the process repeats. Here, the ILP generator follows the same flow by taking the modified KPN (KPN') as an input for the application model.

The KPN captures the communication behavior of the application. We assume applications are implemented using a parallel algorithm, where each task is represented as a process in a KPN. While in a pure KPN, processes are assumed to be deterministic (i.e., the same inputs always produce same outputs), we make no such assumptions, as processes running within our KPN models may maintain internal state. Each node of a KPN corresponds to one process (or task from the parallel algorithm). Communication is accomplished via channels which include unbounded FIFO queues. In a pure KPN, a sender never blocks, as the size of the FIFO is infinite. However, the receiver may block pending data to be sent by the sender. The unbounded nature of the KPN does not present a functional concern when mapped to bounded buffers of the XGRID. Instead, in XGRID the sending processes may be blocked, hence introducing a performance issue rather than a correctness concern. The KPN is annotated with process compute requirements and channel communication requirements. These annotations are obtained from the application and/or algorithm.

The mapping phase optimally maps each process of a KPN to an XGRID core. Moreover, the mapping phase automatically establishes point-to-point communication channels, according to the KPN, by programming appropriate interconnect switches of the XGRID. The mapping problem is formulated as a set of constraints and an objective function in the form of integer linear equations. The objective function, which we are aiming to minimize, is the overall communication cost of the system configuration. We used the CPLEX Optimization Studio 12.3 [196] for solving the set of integer linear equations.

The followings are the constraints and bounds, obtained from the KPN, to create an ILP for the ILP Solver:

$$1 \leq N \leq P$$

$$1 \leq M \leq P$$

where, N and M are the number of the rows and columns in XGRID, respectively and P is the number of the processes in the KPN.

$$P \leq N * M \leq 2 * P$$

The above gives bounds for the number of cores ($N * M$) in XGRID. For all of the followings, i and j are from 0 to P-1.

$$0 \leq Process[i][0] \leq N - 1$$

$$0 \leq Process[i][1] \leq M - 1$$

where, $Process[i][0]$ and $Process[i][1]$ holds row and column values of the target core for process i, respectively. We assume each KPN process is to be mapped onto a single core the premise of XGRID is that cores are plentiful. The result of the following must be different for all processes.

$$Process[i][0] * M + Process[i][1]$$

We use the following equations to calculate a cost function representing communication overhead of the overall system for a mapping solution.

$$Distance_{\text{RD}}[i] = Process[i][1] + 1 \tag{5.1}$$

where, $Distance_{\mathrm{RD}}[i]$ gives the distance between the target core for process i and a serial input unit.

$$Cost_{\mathrm{RD}}[i] = Data_{\mathrm{IN}}[i] * Distance_{\mathrm{RD}}[i] * penalty_{\mathrm{off}} \tag{5.2}$$

where, $Cost_{\mathrm{RD}}[i]$ is the cost of reading data from a serial input unit for process i, $Data_{\mathrm{IN}}[i]$ is the number of read attempts from the serial input unit by process i, $penalty_{\mathrm{off}}$ is a constant penalty for off-chip communication.

$$Distance_{\mathrm{WRT}}[i] = N - Process[i][0] \tag{5.3}$$

where, $Distance_{\mathrm{WRT}}[i]$ gives the distance between the target core for process i and a serial output unit.

$$Cost_{\mathrm{WRT}}[i] = Data_{\mathrm{OUT}}[i] * Distance_{\mathrm{WRT}}[i] * penalty_{\mathrm{off}} \tag{5.4}$$

where, $Cost_{\mathrm{WRT}}[i]$ is the cost of writing data to a serial output unit for process i, $Data_{\mathrm{OUT}}[i]$ is the number of write attempts to the serial output unit by process i.

$$Distance_{\mathrm{COM}}[i][j] = \Big| Process[i][0] - Process[j][0] \Big| + \Big| Process[i][1] - Process[j][1] \Big| \tag{5.5}$$

where, $Distance_{\mathrm{COM}}[i][j]$ gives the distance between the core that process i is mapped to and the core that process j is mapped to.

$$Cost_{\mathrm{COM}}[i][j] = Data_{\mathrm{CHIP}}[i][j] * Distance_{\mathrm{COM}}[i][j] \tag{5.6}$$

where, $Cost_{\mathrm{COM}}[i][j]$ is the cost of on-chip communication between process i and process j, and $Data_{\mathrm{CHIP}}[i][j]$ is the total number of data transfer attempts between process i and process j.

$$Cost_{\mathrm{TOTAL}} = \sum_{i=0}^{P-1} \left\{ Cost_{\mathrm{RD}}[i] + Cost_{\mathrm{WRT}}[i] + \sum_{j=0}^{P-1} Cost_{\mathrm{COM}}[i][j] \right\} \tag{5.7}$$

where, $Cost_{\text{TOTAL}}$ is the global communication cost of the system for a specific mapping solution. The ILP solver (CPLEX) minimizes this total cost function ($Cost_{\text{TOTAL}}$) based on the given constraints and equations.

5.6 System Energy Profiling

Energy consumption is one of the most important characteristics of embedded systems. To evaluate the total energy consumption of XGRID, we take into account both computation and communication energy consumption. The following subsections provide our system energy models.

5.6.1 Computation Energy

A measurement based instruction-level energy model is proposed in [197] by measuring instant current drawn by the target ARM embedded processor and integrating the data to get base energy consumptions. The authors validated their proposed energy model with the experiments, conforming up to 5% error [197]. To profile the energy consumption of XGRID, for each core, we follow an instruction-level energy estimation methodology. The instruction set of XGRID is separated into four categories, namely, arithmetic/logic, load/store, control, and floating point instructions. A constant is defined as a base energy consumption of each category. To calculate these constants, we took averages of base energy consumption of related instructions as described in [197]. Our computation energy formulation is shown below.

$$\mathcal{E}_{\text{compute}} = \sum_{i=0}^{N*M-1} \Big\{ N_{\text{Arth}}[i] * C_1 + N_{\text{LdSt}}[i] * C_2 + N_{\text{Ctrl}}[i] * C_3 + N_{\text{Float}}[i] * C_4 \Big\} \tag{5.8}$$

where, $\mathcal{E}_{\text{compute}}$ is the total energy consumption of a computation, $N * M$ is the number of cores in XGRID; $N_{\text{Arth}}[i]$ is the number of arithmetic/logic instructions for core i; $N_{\text{LdSt}}[i]$ is the number of load/store instructions for core i; $N_{\text{Ctrl}}[i]$ is the number of control instructions for core i; $N_{\text{Float}}[i]$ is the number of floating point instructions for core i; and C_1, C_2, C_3, and C_4 are pre-calculated constants, which can be obtained from gate-level measurements per [197] or instrumentation of a physical core. For our experimentations, and particular XGRID instance, we used 1.328 nJ, 2.368 nJ, 1.644 nJ, and 2.656 nJ for C_1, C_2, C_3, and C_4, respectively.

5.6.2 Communication Energy

In our XGRID framework, there are two types of communication, namely, communication among cores (on-chip communication), and communication between cores and serial high speed I/O units (off-chip communication). We formulate the total energy consumption of communication as the sum of the total energy consumption of on-chip communication and off-chip communication, as shown below.

$$\mathcal{E}_{\text{commun}} = \mathcal{E}_{\text{on-chip}} + \mathcal{E}_{\text{off-chip}} \tag{5.9}$$

where, $\mathcal{E}_{\text{commun}}$ is the total energy consumption of communication, $\mathcal{E}_{\text{on-chip}}$ is the total energy consumption of on-chip communication, and $\mathcal{E}_{\text{off-chip}}$ is the total energy consumption of off-chip communication.

$$\mathcal{E}_{\text{on-chip}} = \sum_{i=0}^{N*M-1} \sum_{j=0}^{N*M-1} \left\{ N_{\text{on-chip}}[i][j] * N_{\text{hop-on}}[i][j] * C_{\text{on-chip}} * {V_{\text{dd}}}^2 \right\} \tag{5.10}$$

where, $N_{\text{on-chip}}[i][j]$ is the total number of on-chip data transfer attempts between core i and core j, $N_{\text{hop-on}}[i][j]$ is the distance between two communicating cores (core i and core

j), $C_{\text{on-chip}}$ is the line capacitance of the channel for on-chip communication, and V_{dd} is the supply voltage. The distance between cores are calculated according to their row and column values (distance = Δrow + Δcolumn).

$$\mathcal{E}_{\text{off-chip}} = \sum_{i=0}^{N*M-1} \left\{ \Big(N_{\text{off-chip-in}}[i] * N_{\text{hop-off-in}}[i] + N_{\text{off-chip-out}}[i] * N_{\text{hop-off-out}}[i] \Big) * C_{\text{off-chip}} * {V_{\text{dd}}}^2 \right\} \tag{5.11}$$

where, $N_{\text{off-chip-in}}[i]$ is the number of read attempts from a serial input unit by core i, $N_{\text{hop-off-in}}[i]$ is the distance between core i and the serial input unit, $N_{\text{off-chip-out}}[i]$ is the number of write attempts to a serial output by core i, $N_{\text{hop-off-out}}[i]$ is the distance between core i and the serial output unit, $C_{\text{off-chip}}$ is the line capacitance of the channel for off-chip communication, and V_{dd} is the supply voltage.

We assume that off-chip data transfers consume more energy than on-chip data transfers. Therefore, we use two distinct constants for the line capacitances. As described in [198], $C_{\text{on-chip}}$ and $C_{\text{off-chip}}$ are set to 1.1 pF and 10 pF, respectively and the supply voltage of XGRID is fixed at 2.7V. These numeric constants will depend on the particular target implementation of the cores and manufacturing technology.

5.7 Deploying XGRID on Thermal Control of a Multi-room Building

With the increasing popularity of CPS approach, the demand for more computational resources has been increasing. Most particularly, large scale CPS applications require data-driven decision making. A large scale CPS can be envisioned as millions of networked smart devices, sensors, and actuators being embedded in the physical world, which can sense, process, and communicate the data all over the network. Such a CPS application benefits from

Algorithm 5.1 A conceptual mapping of control algorithms for the automation of a multi-room building onto target XGRID architecture

```
1: for each core(i,j) in XGRID do
2:   SET_TEMPERATURES();                 ▷ sets initial and reference temperatures
3:   SET_PARAMETERS();                   ▷ sets upper and lower bounds, threshold, etc.
4:     while true do                     ▷ the infinite loop
5:       READ_TEMP_ROOM();               ▷ samples temperature through GPIO
6:       COMPUTE_ROC_ROOM();             ▷ computes the rate of change
7:       if $RoC\_room <= threshold$ then
8:         if $RoC\_room ==$ zero then
9:           NNMT();                     ▷ applies Nearest Neighbor Monitoring Tech.
10:        end if
11:        CTRL_HVAC($temp\_room$);      ▷ controls HVAC operation
12:        WAIT_SEC(36);                 ▷ waits for 36 seconds
13:      else
14:        RBT();                        ▷ applies Read-Back Tech.
15:      end if
16:    end while
17: end for
```

high performance embedded technologies. In this regard, we believe the proposed XGRID architecture has the potential to fulfill compute demands of future CPS applications.

In this section, we provide a conceptual mapping of control algorithms for the automation of a multi-room building onto target XGRID architecture. Algorithm 5.1 shows how each core in XGRID handles temperature control for individual rooms. Each core in XGRID represents a controller deployed in a room. Initial and reference temperatures need to be set by the occupants. The reference temperature refers to the temperature at which the occupants are most comfortable. Initial temperature must be a actual room temperature at the time the temperatures are set for the controller. Upper and lower bounds are determined according to the reference temperature (i.e., set point), as mentioned in Section 4.5.3. The designer may predefine the threshold or have the controller learn it through machine-learning techniques.

RoC_room refers to the rate of change in the room temperature and indicates the temperature difference between two consecutive sensor readings. The reason why the controller waits for 36 seconds is related to sample time. As mentioned in Section 4.7, the controller takes 100

Algorithm 5.2 Procedure to control HVAC operation

```
1: procedure CTRL_HVAC(temperature)
2:     if temperature >= upperbound then INITIATE_COOLER();
3:     else if temperature <= lowerbound then INITIATE_HEATER();
4:     else do nothing;
5:     end if
6: end procedure
```

Algorithm 5.3 Nearest neighbor monitoring procedure for an XGRID core

```
1: procedure NNMT
2:     for each port of core(i,j) in XGRID do
3:         REQUEST_TEMP();                              ▷ Requests temperatures
4:         SELECT_TEMP();                               ▷ Selects temperature
5:     end for
6:     while RoC_room == zero do
7:         temp_neighbor = XPORT(port number);
8:         COMPUTE_ROC_NEIGHBOR();                      ▷ computes RoC for the neighbor
9:         if RoC_neighbor <= threshold and ! = zero then
10:            CTRL_HVAC(temp_neighbor);
11:            WAIT_SEC(36);
12:        else Break;
13:        end if
14:        READ_TEMP_ROOM();                            ▷ samples temperature through GPIO
15:        COMPUTE_ROC_ROOM();                          ▷ computes RoC for the room
16:    end while
17: end procedure
```

samples per hour from the sensor. HVAC is controlled depending on the upper and lower bounds as shown in Algorithm 5.2. If the temperature is above the upper bound then the cooler is initiated by the controller and if it is below the lower bound then the heater is initiated by the controller.

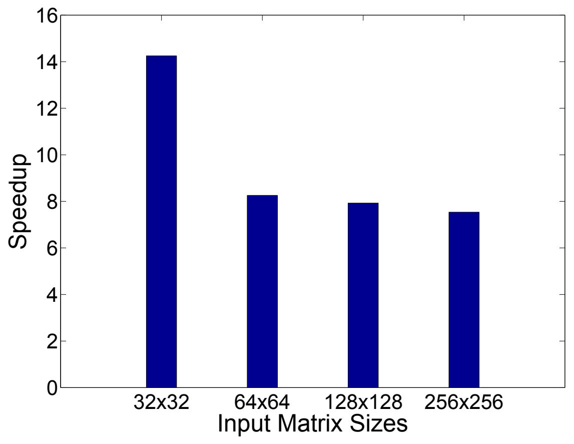
Algorithm 5.3 shows how we apply the nearest neighbor monitoring technique for an XGRID core. As previously mentioned, each core has a number of ports to communicate with other cores. When the rate of change for the room is equal to zero, which indicates the corresponding sensor is stuck, the core requests the temperatures from its neighbors with which it is connected through its ports, and selects the one closest to the previously sampled sensor data. Then selected core becomes the nearest neighbor.

The data taken from the neighbor core is used to compute the rate of change for the neighbor. If the rate of change for the neighbor is below the threshold and not equal to zero, then the temperature provided by the nearest neighbor's sensor is used by the requesting core to control the corresponding HVAC system. Otherwise, the process repeats and another core is selected to become the nearest neighbor core. When the sensor in the room returns to correct behavior, the core stops using the data from the nearest neighbor and uses the temperature data read by the sensor.

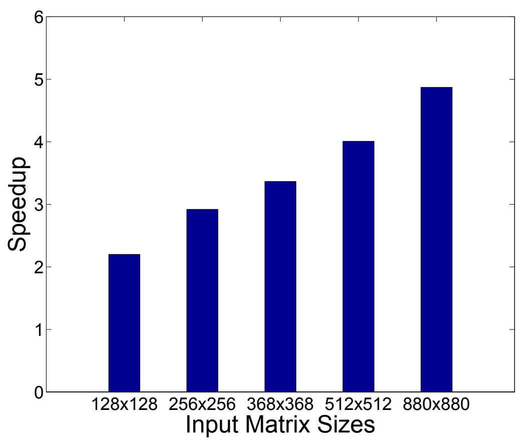
5.8 Experimental Results

We have selected a number of benchmarks to validate the XGRID architecture, performance profiling, and application mapping algorithms. In particular, we have used the 2D DCT (Discrete Cosine Transform) benchmark, MMUL (Matrix Multiplication) benchmark, and four different versions of sorting benchmarks. Sorting algorithm benchmarks consist of the QSORT algorithm and three parallel algorithms based on QSORT, namely, PARALLEL-QSORT [199], HYPER-QSORT [199], and PSRS-QSORT [199].

For each benchmark, we have extracted a KPN and used the ILP approach, presented earlier, to obtain a mapping of the algorithm to our XGRID processor. The specific XGRID processor, used in our experiments, is a 4x4 grid of 32-bit cores, each core having eight 32-bit ports. The communication infrastructure is composed of four 32-bit buses spanning the space between any two adjacent rows of cores. Likewise, the communication infrastructure is composed of four 32-bit buses spanning the space between any two adjacent columns of cores. There are a total of four serial input units, and a total of four serial output units for off-chip communication. Each core has a 1MB data cache and a 64KB instruction cache.



(a)



(b)

Figure 5.5: Speedup (single core vs XGRID) for: (a) 2D DCT, (b) MMUL.

Appendix B contains the details of overall simulation results. Table B.1 and B.2 summarize our simulation results for the 2D DCT running on a single core and many-core, respectively. Table B.3 and B.4 summarize our simulation results for the MMUL running on a single core and many-core, respectively. Table B.6 summarizes our simulation results for the QSORT benchmark running on a single core. Table B.7, B.8, and B.9 summarize our simulation results for the three different parallel versions of QSORT, namely, PARALLEL-QSORT, HYPER-QSORT, and PSRS-QSORT running on many-cores.

Figure 5.5a shows the performance speedup of 2D DCT relative to a single-core implementation for various input sizes. The performance degradation of 2D DCT is expected in case of larger input matrix sizes since I/O wait has considerable effect on the performance for them. On the average, our 16-core XGRID achieved 9X improvement in the performance of DCT. Figure 5.5b shows the performance speedup of MMUL relative to a single-core implementation for various input sizes. On the average, our 16-core XGRID achieved 3X improvement in the performance of MMUL. Moreover, the speedup increased as the input size got larger, as the initial cost of reading the matrices relative to the cost of multiplication diminished.

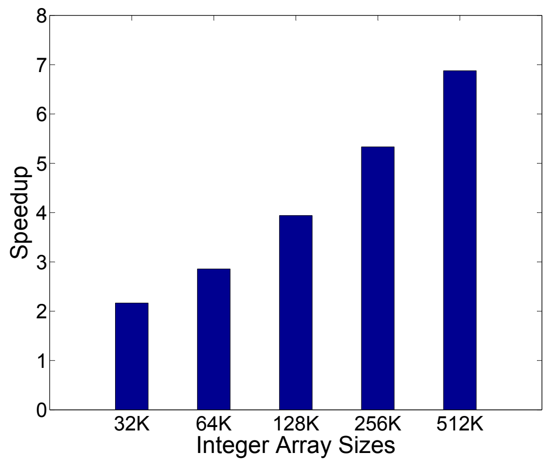
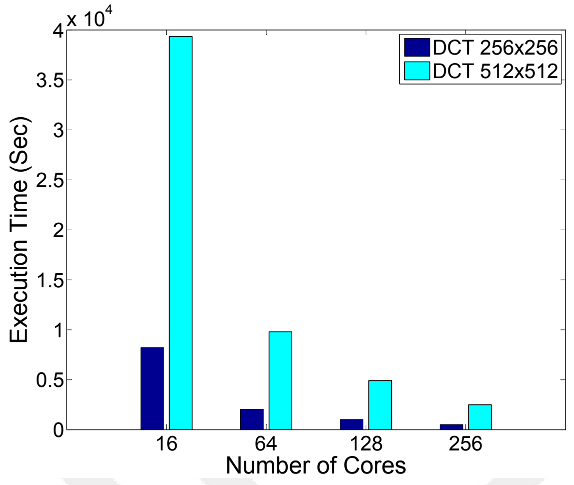


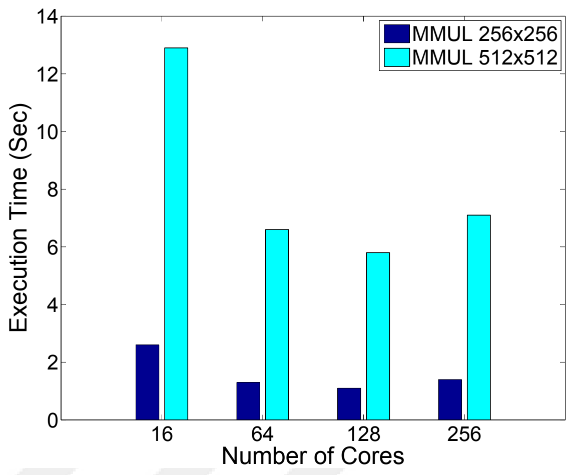
Figure 5.6: Speedup (single core vs XGRID) for PSRS-QSORT.

Figure 5.6 shows the performance speedup of PSRS-QSORT relative to a single-core QSORT implementation. PSRS-QSORT offered the best performance improvement (average of 4X). As with MMUL, the performance improvement of PSRS-QSORT improved, as the size of the input array increased. PARALLEL-QSORT benchmark shows poor performance, compared to HYPER-QSORT and PSRS-QSORT. This is not unexpected, as the latter two versions are optimized for many-core implementations. Execution time of PARALLEL-QSORT is bottlenecked by the last core finishing its execution; hence, the algorithm is likely to do a poor job balancing the number of elements sorted by each core [199]. HYPER-QSORT shows better performance than PARALLEL-QSORT, as expected. Here, the number of elements sorted by each core stays reasonably balanced. As the size of the array to be sorted increases, communication overhead limits scalability of the algorithm [199]. Communication time for HYPER-SORT is dominated by I/O wait. PSRS-QSORT is the best among our parallel algorithms based on QSORT. It does an excellent job balancing the number of elements sorted by each core [199] and I/O wait takes very little time.

We included the results for PARALLEL-QSORT and HYPER-QSORT in Appendix B to point out the importance of efficient parallel programming on manycore architectures. So, the scalability of the different parallel sorting algorithms demonstrates the need for careful algorithm design in many-core implementations.



(a)

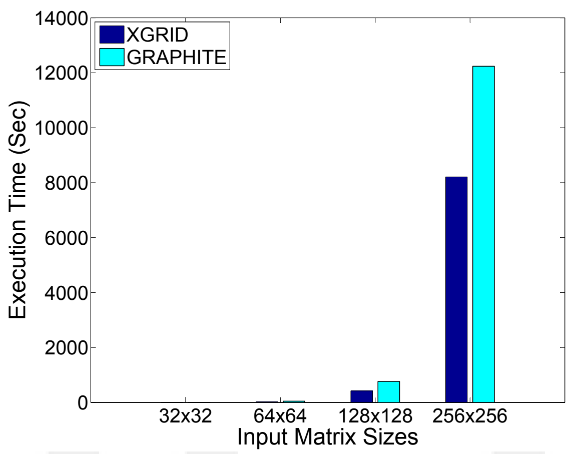


(b)

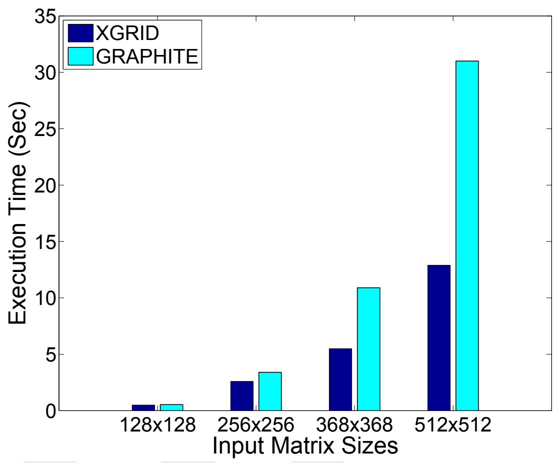
Figure 5.7: Execution time of the benchmarks running on XGRID of various size (core count): (a) 2D DCT, b) MMUL.

In order to validate our scalability claim about XGRID, we ran 2D DCT and MMUL benchmarks on XGRID varying in core counts. Figure 5.7a and Figure 5.7b show the execution time of 2D DCT and MMUL benchmarks on XGRID with different core counts, respectively. 2D DCT benchmarks scale well in all core categories. It is expected since DCT is a computation intense application and the computation dominates the communication in all categories. MMUL benchmarks scale well in all categories except the last one (i.e., 256 cores) where the performance bottleneck occurs. The reason is that matrix multiplication is a communication intense application therefore for increasing number of cores above a certain level, the communication time dominates the computation time and, as a result, causes a decrease in performance. Table B.5 summarizes the results for XGRID varying in core count.

We also validated our assertions on XGRID architecture by comparing XGRID against Graphite many-core architecture. Figure 5.8a and Figure 5.8b show the execution time comparison of 2D DCT and MMUL benchmarks, respectively. For the sake of fairness in comparison, we designated the features stated in Table B.12 for both architectures. We ran some of our benchmarks on an open-source simulator for Graphite many-core architecture [200], to justify the performance of our simulator for XGRID. Graphite integrates a set of



(a)



(b)

Figure 5.8: Execution time comparison of the benchmarks running on 16 cores (XGRID vs GRAPHITE): (a) 2D DCT, b) MMUL.

homogeneous tiles inter-connected by a mesh on-chip network that manages the routing of network packets. Each tile contains a processing core, a memory module, and a network switch [180].

The reasons why we chose the Graphite among existing many-core architectures are that there is an open-source simulator for Graphite and it is well-documented. In addition to that, the Graphite has the tile processor architecture with a mesh on-chip network. Table B.10 summarizes the results for the 2D DCT running on both 16-core XGRID and Graphite. Table B.11 summarizes the results for the MMUL running on both 16-core XGRID and Graphite. The results show that XGRID outperformed Graphite in execution time in all the cases.

5.9 Conclusions

The transition to many-core architectures for embedded and cyber-physical system applications is inevitable because of ever increasing compute demands. With the growing popularity of the cyber-physical systems (CPSs), this increase in compute demands is expected to continue for specialized embedded systems to support a wide range of new applications. In this chapter, we introduced the XGRID many-core embedded processor that makes use of an FPGA-like interconnection network. The XGRID architecture offers numerous advantages, such as low power consumption (due to cores inherently lightweight), hardware supported message passing, and most importantly, scalability as more processing cores are added.

We further described an application mapping algorithm based on Kahn process networks (KPNs) and integer linear programming (ILP) to aid in the mapping of applications on XGRID. We also provided a conceptual mapping of control algorithms for the automation of a multi-room building onto target XGRID architecture as a use case of XGRID in CPS. The building automation application can benefit from many-core embedded processors for temperature control in a distributed fashion or applying controller redundancies for fault mitigation.

Our experimental results are very encouraging. A number of parallel benchmarks are evaluated on XGRID processor using the application mapping technique described in this chapter. Results show an average of 5X speedup, a maximum of 14X speedup, and a minimum of 2X speedup, across all the benchmarks. We observe that, in addition to the need for a scalable architecture, scalable parallel algorithms are required to exploit the compute power of many-core systems. We have validated our scalability claim by running our benchmarks on XGRID varying in core count. We have also validated our assertions on XGRID architecture by comparing XGRID against the Graphite many-core architecture and have shown that XGRID outperforms Graphite in all performance categories.

Chapter 6

Conclusions

The cyber-physical system (CPS) is a term describing a broad range of complex, multi-disciplinary, physically-aware next generation engineered systems that integrate embedded computing technologies (cyber part) into the physical world. CPS is a promising paradigm for the design of current and future engineered systems and is expected to make an important impact on our interactions with the real world. The idea behind CPS places the focus on the integrated system design. Since CPS is a very broad research area, CPSs span diverse applications in different scales. Therefore, each application necessitates strong reasoning capabilities with respect to unique system-level requirements/challenges, the integration of cutting-edge technologies into the related application, and overall impact on the real world.

In this dissertation, the main focus is on sensor fault mitigation and achieving high reliability in CPS operations under faulty sensor conditions. We have proposed a model-based design (MBD) methodology for the design of reliable and fault-tolerant CPSs. Adopting model-based design approach facilitates integrated system development. Since CPSs heavily rely on the data extracted by sensors from the physical phenomena, modeling sensor faults and system evaluation metrics and integrating them into the traditional CPS model become very crucial for conducting system analysis under unexpected conditions (e.g., fault occurrences).

The approaches presented in this study contributes to the design of fault-tolerant CPSs. One of the challenges we have pondered is timely event (e.g., motion as a phenomenon) detection and delivering time-critical operations in CPS under possible faulty sensor conditions. In this regard, our demonstrative example of CPS is the falling ball example (FBE) using binary event detectors (i.e., motion sensors), a controller, and a camera for timely motion detection of a falling ball.

Our findings regarding timely-event detection and delivering time-critical operations show that predictable timing behavior of the system components is very important for reliable CPSs. In fact, many CPS applications are supposed to perform demanded tasks on time, due to continuous and dynamic behavior of the physical world. In this regard, sensor faults and timing characteristics in CPS applications must be analyzed thoroughly and factors leading to the failure of time-critical functions must be addressed clearly.

Another challenge we have pondered is satisfying thermal comfort and energy efficiency under certain faulty sensor conditions in a multi-room building incorporating temperature sensors, controllers, and heating, ventilation, and air conditioning (HVAC) systems as a CPS application. Our findings regarding this challenge show that any occurrence of sensor faults may lead to inefficient use of HVAC systems, elevated energy consumption, and decreased thermal comfort. The analysis of temporal and spatial correlations between sensors' readings to mitigate sensor faults significantly contributes the overall system success by leading to efficient use of HVAC systems, reduced energy consumption, and improved thermal comfort under imperfect sensor conditions.

We have specified well-defined fault semantics for the motion detectors and temperature sensors to make the problem definitions more clear. Our sensor fault models are based on those fault semantics. We have also specified well-defined system evaluation metrics to quantify system success under imperfect conditions. Since the definition of system success is application-specific, system failure and success are defined by the designer according to the mission to be realized.

Our findings regarding reliable CPS design show that the physical system attributes (e.g., sensor placement and environmental effects) might be more dominant than the cyber system attributes. In addition, sensor faults may lead to unsatisfactory system outcome in CPSs since CPSs heavily rely on sensor readings for decision making. Therefore, the analysis of temporal and spatial correlations between sensor readings help mitigate certain type of sensor faults and enable CPSs to utilize sensors' data more efficiently for decision making.

In this dissertation, considering compute demands of large scale CPSs, we also introduced the XGRID many-core embedded processor. XGRID makes use of a novel, FPGA-like, programmable interconnect infrastructure, offering scalability and deterministic communication using hardware supported message passing among cores. The XGRID architecture offers numerous advantages, such as low power consumption (due to cores inherently lightweight), hardware supported message passing, and most importantly, scalability as more processing cores are added.

We have validated our scalability claim by running our benchmarks on XGRID varying in core count. We have also validated our assertions on XGRID architecture by comparing XGRID against the Graphite many-core architecture and have shown that XGRID outperforms Graphite in all performance categories. Furthermore, we provided a conceptual mapping of control algorithms for the automation of a multi-room building onto target XGRID architecture as a use case of XGRID.

Finally, we conclude that the need for domain specificity is a natural consequence of the inherent characteristics of CPSs. Therefore, domain specific analysis of faults and uncertainties (e.g., timing) must be conducted thoroughly for CPS applications. Each CPS application necessitates strong reasoning capabilities with respect to unique system-level requirements/challenges, the integration of cutting-edge technologies into the related application, and overall impact on the real world. Our results indicate that even for not highly complicated CPS applications, the design space is complex and the best system setup is non-trivial. We could demonstrate the effectiveness of our model-based CPS design methodology to simulate the effects of faults on the system outcome by means of the described use cases.

Bibliography

[1] V. Gunes, S. Peter, T. Givargis, and F. Vahid, "A survey on concepts, applications, and challenges in cyber-physical systems," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 8, no. 12, pp. 4242–4268, Dec 2014, © 2014 KSII. Reprinted with permission. [Online]. Available: http://dx.doi.org/10.3837/tiis.2014.12.001

[2] V. Gunes, S. Peter, and T. Givargis, "Modeling and Mitigation of Faults in Cyber-physical Systems with Binary Sensors," in *the 16th International Conference on Computational Science and Engineering*. IEEE, Dec 2013, pp. 515–522, © 2013 IEEE. Reprinted with permission.

[3] V. Gunes, S. Peter, and T. Givargis, "Improving Energy Efficiency and Thermal Comfort of Smart Buildings with HVAC Systems in the Presence of Sensor Faults," in *the 12th International Conference on Embedded Software and Systems (ICESS)*. IEEE, August 2015, © 2015 IEEE. Reprinted with permission.

[4] V. Gunes and T. Givargis, "XGRID: A Scalable Many-Core Embedded Processor," in *the 12th International Conference on Embedded Software and Systems (ICESS)*. IEEE, August 2015, © 2015 IEEE. Reprinted with permission.

[5] R. Baheti and H. Gill, "Cyber-physical systems," *The Impact of Control Technology, IEEE Control Systems Society*, pp. 161–166, 2011.

[6] E. A. Lee, "Cyber Physical Systems: Design Challenges," in *the 11th International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*. IEEE, May 2008, pp. 363–369.

[7] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical Systems: The Next Computing Revolution," in *the 47th Design Automation Conference*. ACM/IEEE, June 2010, pp. 731–736.

[8] E. A. Lee, "CPS Foundations," in *the 47th Design Automation Conference*. ACM/IEEE, June 2010, pp. 737–742.

[9] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of Cyber-Physical Systems," in *the International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, Nov 2011, pp. 1–6.

[10] T. Sanislav and L. Miclea, “Cyber-Physical Systems-Concept, Challenges and Research Areas,” *Journal of Control Engineering and Applied Informatics*, vol. 14, no. 2, pp. 28–33, 2012.

[11] I. Horváth and B. H. Gerritsen, “Cyber-physical systems: Concepts, technologies and implementation principles,” in *the Tools and Methods of Competitive Eng. (TMCE) Symposium, Horváth, I., Rusák, Z., Albers, A., Behrendt, M.(Eds.), Organizing Committee of TMCE*, May 2012, pp. 19–36.

[12] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*, 1st ed. Lee & Seshia, 2011. [Online]. Available: http://leeseshia.org/

[13] “Cyber-Physical Systems Executive Summary,” CPS Steering Group, Tech. Rep., 2008. [Online]. Available: http://iccps.acm.org/2011/_doc/CPS-Executive-Summary.pdf

[14] “Leadership Under Change: Information Technology R&D in a Competitive World,” President’s Council of Advisors on Science and Technology (PCAST), Tech. Rep., 2007. [Online]. Available: https://www.nitrd.gov/Pcast/reports/PCAST-NIT-FINAL.pdf

[15] Cyber-Physical Systems Virtual Organization (CPS-VO). [Online]. Available: http://cps-vo.org/

[16] The Artemis Embedded Computing Systems Initiative. [Online]. Available: http://www.artemis-ju.eu/home_page

[17] The EU Framework Program for Research and Innovation (Horizon2020). [Online]. Available: http://ec.europa.eu/programmes/horizon2020/

[18] E. A. Lee, “Cyber-Physical Systems - Are Computing Foundations Adequate?” in *Position Paper in NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap*, 2006.

[19] P. Marwedel, *Embedded System Design*, 2nd ed. Springer, 2010.

[20] H. Gill, “Cyber-Physical Systems: Beyond ES, SNs, and SCADA,” Presentation in the Trusted Computing in Embedded Systems (TCES) Workshop, 2010. [Online]. Available: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1724&context=sei

[21] L. Mo, X. Cao, J. Chen, and Y. Sun, “Collaborative Estimation and Actuation for Wireless Sensor and Actuator Networks,” in *the 19th World Congress the International Federation of Automatic Control*, Cape Town, South Africa, August 2014, pp. 5544–5549.

[22] Y. Luo, K. Chakrabarty, and T.-Y. Ho, “A cyberphysical synthesis approach for error recovery in digital microfluidic biochips,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, Mar 2012.

[23] T. Springer, S. Peter, and T. Givargis, "Resource Synchronization in Hierarchically Scheduled Real-Time Systems using Preemptive Critical Sections," in *the 10th Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS)*. IEEE, June 2014.

[24] E. Y. Erdem, Y.-M. Chen, M. Mohebbi, J. W. Suh, G. Kovacs, R. B. Darling, and K. F. Böhringer, "Thermally Actuated Omnidirectional Walking Microrobot," *IEEE Journal of Microelectromechanical Systems*, vol. 19, no. 3, pp. 433–442, Jun 2010.

[25] Y. Zhou and J. S. Baras, "CPS Modeling Integration Hub and Design Space Exploration with Application to Microrobotics," *Control of Cyber-Physical Systems, Lecture Notes in Control and Information Sciences, Springer*, vol. 449, pp. 23–42, 2013.

[26] A. A. Hopgood, *Intelligent Systems for Engineers and Scientists*, 3rd ed. CRC Press, 2012.

[27] X. Koutsoukos, N. Kottenstette, J. Hall, P. Antsaklis, and J. Sztipanovits, "Passivity-Based Control Design for Cyber-Physical Systems," in *the International Workshop on Cyber-Physical Systems - Challenges and Applications (CPS-CA)*, 2008.

[28] J. Chen, R. Tan, G. Xing, X. Wang, and X. Fu, "Fidelity-Aware Utilization Control for Cyber-Physical Surveillance Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, pp. 1739–1751, Sep 2012.

[29] D. C. Tarraf, Ed., *Control of Cyber-Physical Systems.* Springer, March 2013, vol. 449.

[30] M. M. Espinosa, "Contributions to the control of networked cyber-physical systems," Ph.D. dissertation, University of California, Los Angeles, 2010.

[31] Radha Poovendran, et al., "A Community Report of the 2008 High Confidence Transportation Cyber-Physical Systems (HCTCPS) Workshop," Tech. Rep., July 2009. [Online]. Available: http://www.ee.washington.edu/research/nsl/aar-cps/NCO_June_2009.pdf

[32] K.-D. Kim and P. R. Kumar, "Cyber-Physical Systems: A Perspective at the Centennial," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, May 2012.

[33] "Cyber Physical Systems: Situation Analysis of Current Trends, Technologies, and Challenges," the National Institute of Standards and Technology (NIST), Tech. Rep., March 2012, prepared by Energetics Incorporated. [Online]. Available: http://events.energetics.com/NIST-CPSWorkshop/pdfs/CPS_Situation_Analysis.pdf

[34] "High-confidence medical devices: Cyber-physical systems for 21st century health care," the Networking and Information Technology Research and Development (NITRD) Program, Tech. Rep., February 2009, prepared by the High Confidence Software and Systems Coordinating Group of NITRD. [Online]. Available: http://www.whitehouse.gov/files/documents/cyber/NITRD-High-ConfidenceMedicalDevices.pdf

[35] R. A. Gupta and M.-Y. Chow, "Networked Control System: Overview and Research Trends," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 7, pp. 2527–2535, Jul 2010.

[36] "Supervisory Control and Data Acquisition (SCADA) systems," National Communications System, Tech. Rep., 2004. [Online]. Available: https://scadahacker.com/library/Documents/ICS_Basics/SCADA%20Basics%20-%20NCS%20TIB%2004-1.pdf

[37] F. Golatowski, J. Blumenthal, M. Handy, M. Haase, H. Burchardt, and D. Timmermann, "Service-Oriented Software Architecture for Sensor Networks," in *the International Workshop on Mobile Computing (IMC)*, vol. 3, 2003, pp. 93–98.

[38] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Global Institute (MGI), Tech. Rep., May 2011. [Online]. Available: http://www.mckinsey.com/~/media/McKinsey/dotcom/Insights%20and%20pubs/MGI/Research/Technology%20and%20Innovation/Big%20Data/MGI_big_data_full_report.ashx

[39] A. Sheth, P. Anantharam, and C. Henson, "Physical-Cyber-Social Computing: An Early 21st Century Approach," *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 78–82, Jan 2013.

[40] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," Recommendations of the National Institute of Standards and Technology, Tech. Rep., September 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[41] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, Jun 2009.

[42] M. Jamshidi, Ed., *System of Systems Engineering: Innovations for the 21st Century*. John Wiley & Sons, November 2008, vol. 58.

[43] T. Samad and T. Parisini, "Systems of Systems," *The Impact of Control Technology, IEEE Control Systems Society*, pp. 175–183, 2011.

[44] W. Bolton, *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*, 5th ed. Pearson Education, 2013.

[45] D. Bradley and D. W. Russell, Eds., *Mechatronics in Action*. Springer, 2010.

[46] N. Wiener, "Cybernetics," *Bulletin of the American Academy of Arts and Sciences*, vol. 3, no. 7, pp. 2–4, Apr 1950.

[47] H. S. Tsien, *Engineering Cybernetics*. McGraw-Hill, 1954.

[48] S. C. Suh, U. J. Tanik, J. N. Carbone, and A. Eroglu, Eds., *Applied Cyber-Physical Systems*. Springer, 2014.

[49] K. Ashton, "That 'Internet of Things' thing," *RFID Journal*, 2009.

[50] R. Caceres and A. Friday, "Ubicomp Systems at 20: Progress, Opportunities, and Challenges," *IEEE Pervasive Computing*, vol. 11, no. 1, pp. 14–21, 2012.

[51] M. Roberti, "The Internet of Things Revisited," *RFID Journal*, 2010.

[52] A. E. Al-Fagih, S. M. A. Oteafy, and H. S. Hassanein, "A pricing scheme for porter based delivery in integrated RFID-Sensor Networks," in *the 37th Annual Conference on Local Computer Networks – Workshops*. IEEE, Oct 2012.

[53] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Elsevier Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct 2010.

[54] A. Koubaa and B. Andersson, "A Vision of Cyber-Physical Internet," in *the 8th International Workshop on Real-Time Networks (RTN)*, 2009.

[55] T. S. Dillon, H. Zhuge, C. Wu, J. Singh, and E. Chang, "Web-of-things framework for cyber-physical systems," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 9, pp. 905–923, Jul 2010.

[56] D. S. Watson, M. A. Piette, O. Sezgen, and N. Motegi, "Machine to machine (M2M) technology in demand responsive commercial buildings," in *the ACEEE 2004 Summer Study on Energy Efficiency in Buildings: Breaking out of the Box*, August 2004, pp. 22–27. [Online]. Available: http://drrc.lbl.gov/publications/machine-machine-m2m-technology-demand

[57] S. Gilani, "The Promise of M2M: How Pervasive Connected Machines are Fueling the Next Wireless Revolution," White Paper, 2009.

[58] M. Chen, J. Wan, and F. Li, "Machine-to-Machine Communications: Architectures, Standards and Applications," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 6, no. 2, pp. 480–497, 2012.

[59] "Disruptive Civil Technologies Six Technologies with Potential Impacts on US Interests out to 2025," The National Intelligence Council, Tech. Rep. CR 2008-07, April 2008. [Online]. Available: http://www.fas.org/irp/nic/disruptive.pdf

[60] S. Paul, J. Pan, and R. Jain, "Architectures for the future networks and the next generation internet: A survey," *Elsevier Computer Communications*, vol. 34, no. 1, pp. 2–42, Jan 2011.

[61] H. Koziolek, R. Weiss, Z. Durdik, J. Stammel, and K. Krogmann, "Towards Software Sustainability Guidelines for Long-living Industrial Systems," in *Software Engineering (Workshops)*. Citeseer, 2011, pp. 47–58.

[62] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0," Final report of the Industrie 4.0 Working Group, Tech. Rep., April 2013. [Online]. Available: http://www.

acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf

[63] "Implementing 21st Century Smart Manufacturing," Smart Manufacturing Leadership Coalition (SMLC) Workshop Summary, Tech. Rep., June 2011. [Online]. Available: https://smartmanufacturingcoalition.org/sites/default/files/implementing_21st_century_smart_manufacturing_report_2011_0.pdf

[64] "Embedded/Cyber-Physical Systems ARTEMIS Major Challenges: 2014-2020," Draft Addendum to the ARTEMIS-SRA 2011, Tech. Rep., December 2013. [Online]. Available: http://www.artemis-ia.eu/publication/download/publication/910/file/ARTEMISIA_SRA_Addendum.pdf

[65] "Industrie 4.0-Smart Manufacturing for the Future," Germany Trade&Invest, Tech. Rep., December 2013. [Online]. Available: http://www.its-owl.de/fileadmin/PDF/News/2014-01-14-Industrie_4.0-Smart_Manufacturing_for_the_Future_German_Trade_Invest.pdf

[66] J. Stankovic, I. Lee, A. Mok, and R. Rajkumar, "Opportunities and obligations for physical computing systems," *Computer*, vol. 38, no. 11, pp. 23–31, 2005.

[67] The SmartAmerica Challenge, A White House Presidential Innovation Fellow project. [Online]. Available: https://www.whitehouse.gov/blog/2014/06/10/smartamerica-challenge-harnessing-power-internet-things

[68] "A report on Critical Infrastructure," The Federal Emergency Management Agency (FEMA) the Strategic Foresight Initiative (SFI), Tech. Rep., June 2011. [Online]. Available: http://www.fema.gov/pdf/about/programs/oppa/critical_infrastructure_paper.pdf

[69] "Crisis Response and Disaster Resilience 2030: Forging Strategic Action in an Age of Uncertainty," The Federal Emergency Management Agency (FEMA), the 2010-2011 Insights of the Strategic Foresight Initiative, Tech. Rep., January 2012. [Online]. Available: http://www.fema.gov/media-library-data/20130726-1816-25045-5167/sfi_report_13.jan.2012_final.docx.pdf

[70] P. A. Bonnefoy, "Scalability of the Air Transportation System and Development of Multi-Airport Systems: A Worldwide Perspective," Ph.D. dissertation, Massachusetts Inst. of Technology, 2008.

[71] "Winning the Future with Science and Technology for 21st Century Smart Systems," Networking and Information Technology Research and Development (NITRD) Program, Tech. Rep. [Online]. Available: http://www.nitrd.gov/nitrdgroups/images/1/12/CPS_OSTP_ResponseWinningTheFuture.pdf

[72] The Federal Aviation Administration (FAA), NextGen. [Online]. Available: http://www.faa.gov/nextgen/

[73] J. A. Momoh, "Fundamentals of analysis and computation for the Smart Grid," *Power and Energy Society (PES) General Meeting*, Jul 2010.

[74] H. Farhangi, "The Path of the Smart Grid," *IEEE Power and Energy Mag.*, vol. 8, no. 1, pp. 18–28, Jan 2010.

[75] "The Smart Grid: An Introduction," The U.S. Department of Energy, Tech. Rep. [Online]. Available: http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/DOE_SG_Book_Single_Pages(1).pdf

[76] I. Lee and O. Sokolsky, "Medical Cyber Physical Systems," in *the 47th Design Automation Conference.* ACM, 2010.

[77] The Center for Integration of Medicine and Innovative Technology, Medical Device Plug-and-Play Interoperability program. [Online]. Available: http://mdpnp.org/

[78] D. Arney, S. Fischmeister, J. M. Goldman, I. Lee, and R. Trausmuth, "Plug-and-Play for Medical Devices: Experiences from a Case Study," *Biomedical Instrumentation & Technology*, vol. 43, no. 4, pp. 313–317, Jul 2009.

[79] J. M. Goldman, "Medical Device CPS Testbeds: Candidate Testbed for Research and Development on Cyber-Physical Medical Device Systems," Presentation in NSF CPS PI Meeting held in Arlington, VA, USA, 2013.

[80] The U.S. Department of Transportation, the Intelligent Transportation Systems (ITS) Program's Research. [Online]. Available: http://www.its.dot.gov/research.htm

[81] F. Qu, F.-Y. Wang, and L. Yang, "Intelligent Transportation Spaces: Vehicles, Traffic, Communications, and Beyond," *IEEE Communications Mag.*, vol. 48, no. 11, pp. 136–142, Nov 2010.

[82] "Cyber-Physical Systems Driving force for innovation in mobility, health, energy and production," Acatech - National Academy of Science and Engineering, Tech. Rep., December 2011. [Online]. Available: http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Publikationen/Stellungnahmen/acatech_POSITION_CPS_Englisch_WEB.pdf

[83] Service Robots-IFR International Federation of Robotics. [Online]. Available: http://www.ifr.org/service-robots/

[84] A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita, and M. Barreto, "Ubiquitous robotics: Recent challenges and future trends," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1162–1172, Nov 2013.

[85] The National Science Foundation (NSF), National Robotics Initiative (NRI). [Online]. Available: http://www.nsf.gov/news/news_summ.jsp?cntn_id=129284

[86] Robotics Virtual Organization (Robotics-VO). [Online]. Available: http://www.robotics-vo.us/

[87] I. Chatzigiannakis, J. P. Drude, H. Hasemann, and A. Kröller, “Developing Smart Homes Using the Internet of Things: How to demonstrate Your System,” *Lecture Notes in Computer Science, Springer*, pp. 415–426, 2014.

[88] C.-U. Lei, K. L. Man, H.-N. Liang, E. G. Lim, and K. Wan, “Building an Intelligent Laboratory Environment via a Cyber-Physical System,” *International Journal of Distributed Sensor Networks*, vol. 2013, pp. 1–9, 2013.

[89] A. GhaffarianHoseini, N. D. Dahlan, U. Berardi, A. GhaffarianHoseini, and N. Makaremi, “The essence of future smart houses: From embedding ICT to adapting to sustainability principles,” *Renewable and Sustainable Energy Reviews*, vol. 24, pp. 593–607, Aug 2013.

[90] W. Meng, R. Ma, and H.-H. Chen, “Smart grid neighborhood area networks: a survey,” *IEEE Network*, vol. 28, no. 1, pp. 24–32, 2014.

[91] K. Wan and V. Alagar, “Dependable Context-Sensitive Services in Cyber Physical Systems,” in *the 10th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, Nov 2011.

[92] G. Denker, N. Dutt, S. Mehrotra, M.-O. Stehr, C. Talcott, and N. Venkatasubramanian, “Resilient dependable cyber-physical systems: a middleware perspective,” *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 41–49, Jan 2012.

[93] K. Höfig, “A vehicle control platform as safety element out of context,” Presentation at HiPEAC Computing Systems Week, Barcelona, Spain, May 2014. [Online]. Available: http://rts.eit.uni-kl.de/hipeac-ws-0514/Presentations/KaiHoefig.pdf

[94] S. Ruiz-Arenas, I. Horváth, R. Mejía-Gutiérrez, and E. Z. Opiyo, “Towards the Maintenance Principles of Cyber-Physical Systems,” *SV - Journal of Mechanical Engineering*, vol. 60, no. 12, pp. 815–831, Dec 2014.

[95] S. A. Haque, S. M. Aziz, and M. Rahman, “Review of Cyber-Physical System in Healthcare,” *International Journal of Distributed Sensor Networks*, vol. 2014, pp. 1–20, 2014.

[96] “Strategic R&D Opportunities for 21st Century Cyber-Physical Systems,” the Steering Committee for Foundations in Innovation for Cyber-Physical Systems, Tech. Rep., January 2013. [Online]. Available: http://www.nist.gov/el/upload/12-Cyber-Physical-Systems020113_final.pdf

[97] M. Rungger and P. Tabuada, “A Notion of Robustness for Cyber-Physical Systems,” *CoRR*, Oct 2013. [Online]. Available: http://arxiv.org/abs/1310.5199

[98] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, “Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber-Physical Systems,” *Proceedings of the IEEE*, vol. 100, no. 1, pp. 283–299, Jan 2012.

[99] R. K. Agarwal, “Review of Technologies to Achieve Sustainable (Green) Aviation,” *InTechOpen Recent Advances in Aircraft Technology*, vol. 19, pp. 427–464, Feb 2012.

[100] S. Misra and E. Eronu, “Implementing Reconfigurable Wireless Sensor Networks: The Embedded Operating System Approach,” *InTechOpen Embedded Systems - High Performance Systems, Applications and Projects*, vol. 11, pp. 221–232, Mar 2012.

[101] J. Scott, A. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar, “PreHeat: Controling home heating using occupancy prediction,” in *the 13th international conference on Ubiquitous computing (UbiComp)*. ACM, 2011.

[102] “Designed-In Cyber Security for Cyber-Physical Systems,” the Cyber Security Research Alliance (CSRA), Gaithersburg, Maryland, Tech. Rep., April 2013. [Online]. Available: http://www.cybersecurityresearch.org/documents/CSRA_Workshop_Report.pdf

[103] R. W. Anwar and S. Ali, “Trust Based Secure Cyber Physical Systems,” in *Workshop Proceedings: Trustworthy Cyber-Physical Systems, Tech Report Series*, no. CS-TR-1347. Computing Science, Newcastle University, September 2012, pp. 9–18.

[104] Y. Mo and B. Sinopoli, “Integrity Attacks on Cyber-Physical Systems,” in *the 1st international conference on High Confidence Networked Systems (HiCoNS)*. ACM, 2012.

[105] G. Loukas, D. Gan, and T. Vuong, “A Review of Cyber Threats and Defence Approaches in Emergency Management,” *Future Internet*, vol. 5, no. 2, pp. 205–236, May 2013.

[106] J. Gertler, “U.S. Unmanned Aerial Systems,” Congressional Research Service Report for Congress, Tech. Rep., January 2012. [Online]. Available: http://fas.org/sgp/crs/natsec/R42136.pdf

[107] “NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 2.0,” the National Institute of Standards and Technology (NIST), Tech. Rep. NIST Special Publication 1108R2, February 2012. [Online]. Available: http://www.nist.gov/smartgrid/upload/NIST_Framework_Release_2-0_corr.pdf

[108] “A Roadmap for Cybersecurity Research,” the U.S. Department of Homeland Security, Tech. Rep., November 2009. [Online]. Available: http://www.dhs.gov/sites/default/files/publications/CSD-DHS-Cybersecurity-Roadmap.pdf

[109] M. J. Park, D. K. Kim, W.-T. Kim, and S.-M. Park, “Dynamic Software Updates in Cyber-Physical Systems,” in *the International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Nov 2010, pp. 425–426.

[110] K. Sampigethaya and R. Poovendran, “Aviation cyber-physical systems: Foundations for future aircraft and air transport,” *Proceedings of the IEEE*, vol. 101, no. 8, pp. 1834–1855, Aug 2013.

[111] M. D. Moore, "Aviation frontiers on-demand aircraft," in *the 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2010.

[112] "Workshop on future directions in cyber-physical systems security," Report on workshop organized by Department of Homeland Security (DHS), Tech. Rep., 2010. [Online]. Available: http://www.ee.washington.edu/faculty/radha/dhs_cps.pdf

[113] R. N. Weber and E. Euteneuer, "Avionics to enable uas integration into the nextgen ats," in *AIAA Guidance, Navigation, and Control Conference*, 2010.

[114] D. U. Keerti K. Bhamidipati and N. Neogi, "Engineering safety and reliability into uav systems: Mitigating the ground impact hazard," in *AIAA Guidance, Navigation, and Control Conference*, 2007.

[115] A. A. Cárdenas and R. Safavi-Naini, "Chapter 25 - security and privacy in the smart grid," in *Handbook on Securing Cyber-Physical Critical Infrastructure*, S. K. Das, K. Kant, and N. Zhang, Eds. Boston: Morgan Kaufmann, 2012, pp. 637 – 654.

[116] J. Liu, Y. Xiao, S. Li, W. Liang, and C. L. P. Chen, "Cyber security and privacy issues in smart grids," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, pp. 981–997, Fourth Quarter 2012.

[117] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Elsevier Computer Networks*, vol. 57, no. 5, pp. 1344 – 1371, 2013.

[118] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber-physical system security for the electric power grid," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 210–224, Jan 2012.

[119] A. Dominguez-Garcia, "Reliability modeling of cyber-physical electric power systems: A system-theoretic framework," in *the IEEE Power and Energy Society General Meeting*, July 2012, pp. 1–5.

[120] A. Rohmer, "Emergency Response in Large-Scale Disasters: Lessons Learned and Implications for National Security," *Government Honors Papers*, p. 9, 2010. [Online]. Available: http://digitalcommons.conncoll.edu/govhp/9/

[121] J. Treglia, L. McKnight, A. Kuehn, A. Ramnarine-Rieks, M. Venkatesh, and T. Bose, "Interoperability by 'edgeware': Wireless grids for emergency response," in *44th Hawaii International Conference on System Sciences (HICSS)*, Jan 2011, pp. 1–10.

[122] J. Andersson, R. de Lemos, S. Malek, and D. Weyns, "Modeling dimensions of self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems*, ser. Lecture Notes in Computer Science, B. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Springer Berlin Heidelberg, 2009, vol. 5525, pp. 27–47.

[123] J. Marsden, J. Treglia, and L. McKnight, "Dynamic emergency response communication: The intelligent deployable augmented wireless gateway (idawg)," in *the International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. IEEE, March 2012, pp. 279–286.

[124] “Smart, Safe, and Sustainable Manufacturing,” Rockwell Automation 2013 Corporate Responsibility Report, Tech. Rep., 2013. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/br/esap-br018_-en-p.pdf

[125] D. Zühlke and L. Ollinger, “Agile automation systems based on cyber-physical systems and service-oriented architectures,” in *Advances in Automation and Robotics, Vol.1*, ser. Lecture Notes in Electrical Engineering, G. Lee, Ed. Springer Berlin Heidelberg, 2012, vol. 122, pp. 567–574.

[126] J. Hur and K. Kang, “Dependable and secure computing in medical information systems,” *Elsevier Computer Communications*, vol. 36, no. 1, pp. 20 – 28, 2012.

[127] M.-W. Jung and J. Cho, “Interoperability and control systems for medical cyber physical systems,” in *IT Convergence and Security 2012*, ser. Lecture Notes in Electrical Engineering, K. J. Kim and K.-Y. Chung, Eds. Springer Netherlands, 2013, vol. 215, pp. 283–291.

[128] F. Koushanfar, A.-R. Sadeghi, and H. Seudie, “Eda for secure and dependable cybercars: Challenges and opportunities,” in *the 49th Annual Design Automation Conference.* ACM, 2012, pp. 220–228.

[129] P. Patil, “Towards reliable communication in intelligent transportation systems,” in *the 31st Symposium on Reliable Distributed Systems (SRDS).* IEEE, Oct 2012, pp. 485–486.

[130] P. F. Santana, J. Barata, and L. Correia, “Sustainable robots for humanitarian demining,” *International Journal of Advanced Robotics Systems (ARS-journal)*, vol. 4, no. 2, pp. 207–218, 2007.

[131] B. Scholz-Reiter, M. Rohde, S. Kunaschk, and M. Lütjen, “Towards automation of low standardized logistic processes by use of cyber physical robotic systems (cprs),” in *the 13th WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering.* World Scientific and Engineering Academy and Society (WSEAS), 2011, pp. 293–298.

[132] J. Park, I. Chun, H. Lee, W.-T. Kim, and E. Lee, “Intelligent service robot and application operating in cyber-physical environment,” in *Embedded and Multimedia Computing Technology and Service*, ser. Lecture Notes in Electrical Engineering, J. J. J. H. Park, Y.-S. Jeong, S. O. Park, and H.-C. Chen, Eds. Springer Netherlands, 2012, vol. 181, pp. 301–310.

[133] Z. Ye, H. Corporaal, and P. Jonker, “Phd forum: A cyber-physical system approach to embedded visual servoing,” in *the 5th ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, Aug 2011, pp. 1–2.

[134] S. Peter, F. Vahid, D. D. Gajski, and T. Givargis, “A Ball Goes to School-Our Experiences from a CPS Design Experiment,” in *1st Workshop on Cyber-Physical Systems Education (CPS-Ed)*, Apr 2013.

[135] J. Sun, Y. Zhang, B. Wang, and K. He, “Providing Timely Response in Smart Car,” *Computer, Informatics, Cybernetics and Applications, Lecture Notes in Electrical Engineering, Springer*, vol. 107, pp. 211–220, 2012.

[136] J. Schiff and K. Goldberg, “Automated Intruder Tracking using Particle Filtering and a Network of Binary Motion Sensors,” in *the International Conference on Automation Science and Engineering.* IEEE, Oct 2006, pp. 580–587.

[137] B. Johnson and H. Kress-Gazit, “Probabilistic analysis of correctness of high-level robot behavior with sensor error,” in *Robotics: Science and Systems Conference*, June 2011.

[138] L.-A. Tang, X. Yu, S. Kim, J. Han, C.-C. Hung, and W.-C. Peng, “Tru-Alarm: Trustworthiness Analysis of Sensor Networks in Cyber-Physical Systems,” in *the 10th International Conference on Data Mining (ICDM).* IEEE, Dec 2010, pp. 1079–1084.

[139] Z. Wang, E. Bulut, and B. Szymanski, “A Distributed Cooperative Target Tracking with Binary Sensor Networks,” in *the International Conference on Communications Workshops (ICC).* IEEE, May 2008, pp. 306–310.

[140] D. De, W.-Z. Song, M. Xu, C.-L. Wang, D. Cook, and X. Huo, “FindingHuMo: Real-Time Tracking of Motion Trajectories from Anonymous Binary Sensing in Smart Environments,” in *the 32nd International Conference on Distributed Computing Systems (ICDCS).* IEEE, June 2012, pp. 163–172.

[141] D. Willner, C. Chang, and K. Dunn, “Kalman filter algorithms for a multi-sensor system,” in *Conference on Decision and Control including the 15th Symposium on Adaptive Processes.* IEEE, Dec 1976, pp. 570–574.

[142] X. Wang, N. Hovakimyan, and L. Sha, “L1simplex: Fault-tolerant control of cyber-physical systems,” in *the 4th International Conference on Cyber-Physical Systems.* ACM/IEEE, 2013, pp. 41–50.

[143] E. Kim and N. Shanbhag, “Soft n-modular redundancy,” *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 323–336, March 2012.

[144] Y. Yeh, “Triple-triple redundant 777 primary flight computer,” in *Aerospace Applications Conference*, vol. 1. IEEE, Feb 1996, pp. 293–307.

[145] D. Blough and G. Sullivan, “A comparison of voting strategies for fault-tolerant distributed systems,” in *the 9th Symposium on Reliable Distributed Systems*, Oct 1990, pp. 136–145.

[146] S. Neema, T. Bapty, S. Shetty, and S. Nordstrom, “Autonomic fault mitigation in embedded systems,” *Elsevier Engineering Applications of Artificial Intelligence*, vol. 17, no. 7, pp. 711 – 725, 2004.

[147] M. Daigle and K. Goebel, “Model-based prognostics under limited sensing,” in *Aerospace Conference.* IEEE, March 2010, pp. 1–12.

[148] R. Isermann, R. Schwarz, and S. Stolzl, "Fault-tolerant drive-by-wire systems," *IEEE Control Systems*, vol. 22, no. 5, pp. 64–81, Oct 2002.

[149] H. Moneva, R. Hamberg, and T. Punter, "A Design Framework for Model-based Development of Complex Systems," in *the 32nd Real-Time Systems Symposium, the 2nd Analytical Virtual Integration of Cyber-Physical Systems Workshop.* IEEE, 2011.

[150] D. Shang, E. Eyisi, Z. Zhang, X. Koutsoukos, J. Porter, G. Karsai, and J. Sztipanovits, "A case study on the model-based design and integration of automotive cyber-physical systems," in *the 21st Mediterranean Conference on Control Automation (MED)*, June 2013, pp. 483–492.

[151] D. Broman, "High-confidence Cyber-physical Co-design," *ACM SIGBED Review*, vol. 10, no. 2, pp. 23–23, July 2013.

[152] J. Jensen, D. Chang, and E. Lee, "A model-based design methodology for cyber-physical systems," in *the 7th International Wireless Communications and Mobile Computing Conference (IWCMC)*, July 2011, pp. 1666–1671.

[153] MATLAB R2013a, Mathworks. [Online]. Available: http://www.mathworks.com/products/matlab/

[154] Simulink in R2013a, Mathworks. [Online]. Available: http://www.mathworks.com/products/simulink/

[155] Air and Radiation Basic Information. [Online]. Available: http://www.epa.gov/air/basic.html

[156] "Temperature Wars: Savings vs. Comfort," International Facility Management Association (IFMA), Tech. Rep. [Online]. Available: http://www.ifma.org/docs/surveys/hvacsurvey2009.pdf?sfvrsn=2

[157] V. Reppa, P. Papadopoulos, M. Polycarpou, and C. Panayiotou, "A Distributed Architecture for HVAC Sensor Fault Detection and Isolation," *IEEE Transactions on Control Systems Technology*, vol. PP, no. 99, pp. 1–1, 2014.

[158] S. Wang, Q. Zhou, and F. Xiao, "A system-level fault detection and diagnosis strategy for HVAC systems involving sensor faults," *Elsevier Energy and Buildings*, vol. 42, no. 4, pp. 477 – 490, 2010.

[159] Z. Du, B. Fan, X. Jin, and J. Chi, "Fault detection and diagnosis for buildings and HVAC systems using combined neural networks and subtractive clustering analysis," *Elsevier Building and Environment*, vol. 73, no. 0, pp. 1 – 11, 2014.

[160] X.-B. Yang, X.-Q. Jin, Z.-M. Du, Y.-H. Zhu, and Y.-B. Guo, "A hybrid model-based fault detection strategy for air handling unit sensors," *Elsevier Energy and Buildings*, vol. 57, no. 0, pp. 132 – 143, 2013.

[161] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor Faults: Detection Methods and Prevalence in Real-world Datasets," *ACM Transactions on Sensor Networks*, vol. 6, no. 3, pp. 23:1–23:39, June 2010.

[162] K. Ni *et al.*, "Sensor Network Data Fault Types," *ACM Transactions on Sensor Networks*, vol. 5, no. 3, pp. 25:1–25:29, May 2009.

[163] H. Klee and R. Allen, *Simulation of Dynamic Systems with MATLAB® and Simulink®*, 2nd ed. CRC Press, 2011.

[164] "Electric Power Monthly with Data for January 2015," the U.S. Energy Information Administration (EIA), Tech. Rep., 2015. [Online]. Available: http://www.eia.gov/electricity/monthly/pdf/epm.pdf

[165] H. Sutter, "The free lunch is over: A fundamental turn toward concurrency in software," *Dr. Dobb's journal*, vol. 30, no. 3, pp. 202–210, March 2005.

[166] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware Microarchitecture," in *the 30th Annual International Symposium on Computer Architecture.* ACM, 2003, pp. 2–13.

[167] S. Borkar, "Thousand Core Chips: A Technology Perspective," in *the 44th Annual Design Automation Conference.* ACM, 2007, pp. 746–749.

[168] A. Kayi, T. El-Ghazawi, and G. B. Newby, "Performance issues in emerging homogeneous multi-core architectures," *Simulation Modelling Practice and Theory*, vol. 17, no. 9, pp. 1485 – 1499, 2009, advances in System Performance Modelling, Analysis and Enhancement.

[169] A. Baumann *et al.*, "The Multikernel: A New OS Architecture for Scalable Multicore Systems," in *the 22nd Symposium on Operating Systems Principles.* ACM SIGOPS, 2009, pp. 29–44.

[170] S. Pasricha and N. Dutt, *On-chip communication architectures: system on chip interconnect.* Morgan Kaufmann, 2010.

[171] W. Zhang, G.-M. Du, Y. Xu, M.-L. Gao, L.-F. Geng, B. Zhang, Z.-Y. Jiang, N. Hou, and Y.-H. Tang, "Design of a Hierarchy-Bus Based MPSoC on FPGA," in *the 8th International Conference on Solid-State and Integrated Circuit Technology.* IEEE, 2006, pp. 1966–1968.

[172] E. Carara, R. de Oliveira, N. Calazans, and F. Moraes, "HeMPS - a framework for NoC-based MPSoC generation," in *the International Symposium on Circuits and Systems (ISCAS).* IEEE, May 2009, pp. 1345–1348.

[173] S. Tota, M. Casu, M. Ruo Roch, L. Macchiarulo, and M. Zamboni, "A Case Study for NoC-Based Homogeneous MPSoC Architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 3, pp. 384–388, March 2009.

[174] S. Tota, M. Casu, M. Ruo Roch, L. Rostagno, and M. Zamboni, “MEDEA: a hybrid shared-memory/message-passing multiprocessor NoC-based architecture,” in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2010, pp. 45–50.

[175] J. Cong, K. Gururaj, G. Han, A. Kaplan, M. Naik, and G. Reinman, “MC-Sim: An Efficient Simulation Tool for MPSoC Designs,” in *the International Conference on Computer-Aided Design*. IEEE, 2008, pp. 364–371.

[176] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, “The M5 simulator: Modeling networked systems,” *IEEE Micro*, vol. 26, no. 4, pp. 52–60, 2006.

[177] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, “Multifacet’s General Execution-driven Multiprocessor Simulator (GEMS) Toolset,” *SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92–99, Nov 2005.

[178] N. Binkert *et al.*, “The Gem5 Simulator,” *SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, Aug 2011.

[179] P. Ren, M. Lis, M. H. Cho, K. S. Shim, C. Fletcher, O. Khan, N. Zheng, and S. Devadas, “HORNET: A Cycle-Level Multicore Simulator,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 6, pp. 890–903, June 2012.

[180] J. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, “Graphite: A distributed parallel simulator for multicores,” in *The 16th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Jan 2010, pp. 1–12.

[181] S. Bell *et al.*, “TILE64 - Processor: A 64-Core SoC with Mesh Interconnect,” in *the International Solid-State Circuits Conference (ISSCC)*. IEEE, Feb 2008, pp. 88–598.

[182] 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 *the 42nd International Symposium on Microarchitecture*. IEEE, Dec 2009, pp. 469–480.

[183] P. Guerrier and A. Greiner, “A Generic Architecture for On-chip Packet-switched Interconnections,” in *Design, Automation and Test in Europe (DATE) Conference*. ACM, 2000, pp. 250–256.

[184] L. Benini and G. De Micheli, “Networks on chips: a new SoC paradigm,” *IEEE Transactions on Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[185] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, “Performance evaluation and design trade-offs for network-on-chip interconnect architectures,” *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, Aug 2005.

[186] U. Y. Ogras, J. Hu, and R. Marculescu, “Key Research Problems in NoC Design: A Holistic Perspective,” in *the 3rd International Conference on Hardware/Software Codesign and System Synthesis*. IEEE/ACM, 2005, pp. 69–74.

[187] S. Patel, P. Parandkar, S. Katiyal, and A. Agrawal, “Exploring Alternative Topologies for Network-on-Chip Architectures,” *Bharati Vidyapeeth’s Institute of Computer Applications and Management (BVICAM) International Journal of Information Technology (BIJIT)*, vol. 3, no. 2, pp. 372–376, Aug 2011.

[188] T. Simunic, S. Boyd, and P. Glynn, “Managing power consumption in networks on chips,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 1, pp. 96–107, Jan 2004.

[189] J. W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, “Congestion-controlled Best-effort Communication for Networks-on-chip,” in *Design, Automation and Test in Europe (DATE) Conference*. ACM EDA Consortium, 2007, pp. 948–953.

[190] E. Kusse and J. Rabaey, “Low-energy embedded FPGA structures,” in *The International Symposium on Low Power Electronics and Design*, Aug 1998, pp. 155–160.

[191] L. Shang, A. S. Kaviani, and K. Bathala, “Dynamic Power Consumption in Virtex-II FPGA Family,” in *the 10th International Symposium on Field-programmable Gate Arrays*. ACM, 2002, pp. 157–164.

[192] K. K. W. Poon, S. J. E. Wilton, and A. Yan, “A Detailed Power Model for Field-programmable Gate Arrays,” *ACM Trans. Design Automation of Electronic Systems*, vol. 10, no. 2, pp. 279–302, Apr 2005.

[193] D. W. Walker, “The design of a standard message passing interface for distributed memory concurrent computers,” *Parallel Computing*, vol. 20, no. 4, pp. 657–673, April 1994.

[194] LCC, A Retargetable C Compiler. [Online]. Available: https://aur.archlinux.org/packages/lcc-compiler/

[195] K. Gilles, “The semantics of a simple language for parallel programming,” in *the IFIP Congress In Information Processing*, vol. 74, 1974, pp. 471–475.

[196] CPLEX Optimization Studio. [Online]. Available: http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud

[197] N. Kavvadias, P. Neofotistos, S. Nikolaidis, C. Kosmatopoulos, and T. Laopoulos, “Measurements analysis of the software-related power consumption in microprocessors,” *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 4, pp. 1106–1112, Aug 2004.

[198] A. Abril Garcia, J. Gobert, T. Dombek, H. Mehrez, and F. Petrot, “Cycle-accurate energy estimation in system level descriptions of embedded systems,” in *the 9th International Conference on Electronics, Circuits and Systems*. IEEE, 2002, pp. 549–552.

[199] M. J. Quinn, *Parallel programming in C with MPI and OpenMP.* McGraw-Hill Higher Education, 2004.

[200] Graphite Simulator Source Code. [Online]. Available: https://github.com/mit-carbon/Graphite/wiki

Appendix A

Details of Experiment Results for Chapter 4

Table A.1: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts for the perfect case without fault occurrences.

Place	**Energy Consumption (kWh)**			**Cost ($)**	**Thermal Comfort**
	Cooler	**Heater**	**Total**		
Room1	3.89	11.42	15.31	2.68	100%
Room2	1.91	7.58	9.49	1.66	100%
Room3	2.99	10.94	13.93	2.43	100%
Building	8.79	29.94	38.73	6.77	100%

Table A.2: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when single-sample-spike (SSS) faults with positive spike occur in Room2 and are not mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**
	Cooler	**Heater**	**Total**			
Room1	4.1	11.58	15.68	+ 2.4%	2.75	100%
Room2	2.27	7.76	10.03	+ 5.6%	1.76	100%
Room3	2.99	10.73	13.72	- 1.5%	2.4	100%
Building	9.36	30.07	39.43	+ 1.8%	6.91	100%

Table A.3: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when single-sample-spike (SSS) faults with positive spike occur in Room2 and are mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**	**Mitigate Success**
	Cooler	**Heater**	**Total**				
Room1	3.89	11.42	15.31	0%	2.68	100%	100%
Room2	1.91	7.58	9.49	0%	1.66	100%	
Room3	3	10.93	13.93	0%	2.43	100%	
Building	8.8	29.93	38.73	0%	6.77	100%	

Table A.4: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when single-sample-spike (SSS) faults with negative spike occur in Room2 and are not mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**
	Cooler	**Heater**	**Total**			
Room1	4.1	11.44	15.54	+ 1.5%	2.72	100%
Room2	2.08	7.82	9.9	+ 4.3%	1.73	100%
Room3	2.99	10.75	13.74	- 1.4%	2.4	100%
Building	9.17	30.01	39.18	+ 1%	6.85	100%

Table A.5: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when single-sample-spike (SSS) faults with negative spike occur in Room2 and are mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**	**Mitigate Success**
	Cooler	**Heater**	**Total**				
Room1	3.89	11.42	15.31	0%	2.68	100%	100%
Room2	1.91	7.58	9.49	0%	1.66	100%	
Room3	2.99	10.94	13.93	0%	2.43	100%	
Building	8.79	29.94	38.73	0%	6.77	100%	

Table A.6: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when spike-and-stay (SAS) faults with positive spike occur in Room2 and are not mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**
	Cooler	**Heater**	**Total**			
Room1	3.86	11.84	15.7	+ 2.5%	2.75	100%
Room2	10.1	9.2	19.3	+ 103%	3.37	40.7%
Room3	3.02	10.57	13.59	- 2.4%	2.38	100%
Building	17.07	31.61	48.68	+ 26%	8.5	80%

Table A.7: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when spike-and-stay (SAS) faults with positive spike occur in Room2 and are mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**	**Mitigate Success**
	Cooler	**Heater**	**Total**				
Room1	3.92	11.65	15.57	+ 1.7%	2.73	100%	97%
Room2	2.31	8.8	11.11	+ 17%	1.94	93.2%	
Room3	3	10.75	13.75	- 1.3%	7.34	100%	
Building	9.23	31.2	40.43	+ 4%	7.07	98%	

Table A.8: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when spike-and-stay (SAS) faults with negative spike occur in Room2 and are not mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**
	Cooler	**Heater**	**Total**			
Room1	3.89	11.77	15.66	+ 2.2%	2.78	100%
Room2	6.2	18.92	25.12	+ 164%	4.39	17.6%
Room3	3.23	10.65	13.88	- 0.4%	2.43	100%
Building	13.32	41.34	54.66	+ 41%	9.6	73%

Table A.9: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when spike-and-stay (SAS) faults with negative spike occur in Room2 and are mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**	**Mitigate Success**
	Cooler	**Heater**	**Total**				
Room1	3.92	11.47	15.39	+ 0.5%	2.7	100%	98.3%
Room2	2.25	8.06	10.31	+ 8.6%	1.8	96.2%	
Room3	3.02	10.73	13.75	- 1.3%	2.41	100%	
Building	9.19	30.26	39.45	+ 2%	6.91	98.6%	

Table A.10: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when stuck-at (SA) faults occur in Room2 and are not mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**
	Cooler	**Heater**	**Total**			
Room1	3.89	11.79	15.68	+ 2.4%	2.74	100%
Room2	1.9	5.49	7.39	- 22.1%	1.29	65.1%
Room3	2.99	10.59	13.58	- 2.5%	2.37	100%
Building	8.78	27.87	36.65	- 5.3%	6.4	88.3%

Table A.11: Results at the end of 24 hours that show energy consumptions, costs, and thermal comforts when stuck-at (SA) faults occur in Room2 and are mitigated.

Place	**Energy Consumption (kWh)**			**Variation in Energy**	**Cost ($)**	**Thermal Comfort**	**Mitigate Success**
	Cooler	**Heater**	**Total**				
Room1	3.89	11.68	15.57	+ 1.7%	2.72	100%	98.25%
Room2	1.9	8.96	10.86	+ 14.4%	1.9	100%	
Room3	2.99	10.7	13.69	- 1.7%	2.39	100%	
Building	8.78	31.34	40.12	+ 3.5%	7.01	100%	

Appendix B

Details of Experiment Results for Chapter 5

Table B.1: Execution time and energy consumption of 2D-DCT for single core.

2D-DCT Matrix Size	**Execution Time (s)**	**Total Energy (J)**	**Total Energy of Computation (J)**	**Total Energy of Communication (µJ)**
32 x 32	11.4	25.2	25.2	2.2
64 x 64	180	117	117	9
128 x 128	3337	250.2	250.2	36.4
256 x 256	61880	423	423	150

Table B.2: Execution time and energy consumption of 2D-DCT for 16 cores in XGRID.

2D-DCT Matrix Size	**Execution Time (s)**	**Total Energy (J)**	**Total Energy of Computation (J)**	**Total Energy of Communication (µJ)**
32 x 32	0.8	5.6	5.6	2.8
64 x 64	21.8	96.7	96.7	11.3
128 x 128	421	227.6	227.6	46
256 x 256	8210	256	256	188

Table B.3: Execution time and energy consumption of MMUL for single core.

MMUL Matrix Size	Execution Time (s)	Total Energy (J)	Total Energy of Computation (J)	Total Energy of Communication (µJ)
128 x 128	1.1	1	1	35
256 x 256	7.3	7.5	7.5	144
368 x 368	18.5	20	20	287
512 x 512	51.3	53.5	53.5	584
880 x 880	245	62	62	1817

Table B.4: Execution time and energy consumption of MMUL for 16 cores in XGRID.

MMUL Matrix Size	Execution Time (s)	Total Energy (J)	Total Energy of Computation (J)	Total Energy of Communication (µJ)
128 x 128	0.5	0.3	0.3	53
256 x 256	2.5	1.7	1.7	215.5
368 x 368	5.5	4.3	4.3	428.5
512 x 512	12.8	11.5	11.5	870
880 x 880	50.3	53.2	53.2	2674

Table B.5: Execution time of benchmarks running on XGRID of various sizes (core count).

Cores	Execution Time (s)			
	2D DCT		MMUL	
	256x256	512x512	256x256	512x512
16	8210	39351	2.5	12.8
64	2059	9797	1.3	6.6
128	1033	4916	1.1	5.8
256	525	2497	1.4	7.1

Table B.6: Execution time and energy consumption of QSORT for single core.

QSORT Integer Array Size	**Execution Time (s)**	**Total Energy (J)**	**Total Energy of Computation (J)**	**Total Energy of Communication (µJ)**
32K	1.3	1.5	1.5	44.3
64K	4	4	4	89.5
128K	13	14	14	182
256K	46.4	49	49	372
512K	174	99.5	99.5	752

Table B.7: Execution time and energy consumption of PARALLEL-QSORT[199] in case of using 16 cores in XGRID.

PARALLEL Integer Array Size	**Execution Time (s)**	**Total Energy (J)**	**Total Energy of Computation (J)**	**Total Energy of Communication (µJ)**
32K	6.6	3.9	3.9	46.7
64K	33	18.5	18.5	94.4
128K	130	58.5	58.5	192
256K	512	113	113	391.5
512K	2041	190	190	791

Table B.8: Execution time and energy consumption of HYPER-QSORT[199] in case of using 16 cores in XGRID.

HYPER Integer Array Size	**Execution Time (s)**	**Total Energy (J)**	**Total Energy of Computation (J)**	**Total Energy of Communication (µJ)**
32K	1.8	2.7	2.7	46.7
64K	5.8	10	10	94.3
128K	20.6	39.6	39.6	191.6
256K	77.5	149.8	149.8	391
512K	299	220	220	790.5

Table B.9: Execution time and energy consumption of PSRS-QSORT[199] in case of using 16 cores in XGRID.

PSRS Integer Array Size	**Execution Time (s)**	**Total Energy (J)**	**Total Energy of Computation (J)**	**Total Energy of Communication (µJ)**
32K	0.6	0.3	0.3	47
64K	1.4	1	1	94.6
128K	3.3	3.1	3.1	192.3
256K	8.7	10.8	10.8	392.7
512K	25.3	40	40	793.4

Table B.10: XGRID vs GRAPHITE[180] executing 2D DCT on 16 cores.

Input Size	Execution Time (s)		Improvement
	XGRID	GRAPHITE	
32x32	0.8	3.0	275%
64x64	21.8	47.6	118%
128x128	421	764.2	82%
256x256	8212	12236	49%

Table B.11: XGRID vs GRAPHITE[180] executing MMUL on 16 cores.

Input Size	Execution Time (s)		Improvement
	XGRID	GRAPHITE	
128x128	0.5	0.54	8%
256x256	2.6	3.4	31%
368x368	5.5	10.9	98%
512x512	12.9	31	140%

Table B.12: The designated values for some features of XGRID and GRAPHITE[180] architectures.

Feature	XGRID	GRAPHITE
Number of cores	16	16
Clock frequency	130 Mhz	130 Mhz
Instruction cache	Private, 64 Kb (per core)	Private, 64 Kb (per tile)
Data cache	Private, 1 Mb (per core)	Private, 1 Mb (per tile)
Interconnect topology	2D grid network	2D mesh network