Uncertainty Aware Mapping of Embedded Systems for Reliability, Performance, and Energy

Wenkai Guan
Marquette University

UNCERTAINTY AWARE MAPPING OF EMBEDDED SYSTEMS FOR RELIABILITY, PERFORMANCE, AND ENERGY

by

Wenkai Guan

A Thesis Submitted to the Faculty of the Graduate School
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

May 2018

ABSTRACT

UNCERTAINTY AWARE MAPPING OF EMBEDDED SYSTEMS FOR RELIABILITY, PERFORMANCE, AND ENERGY

Wenkai Guan
Marquette University

Due to technology downscaling, embedded systems have increased in complexity and heterogeneity. Increasingly large process, voltage, and temperature variations negatively affect the design and optimization process of these systems. These factors contribute to increased uncertainties that in turn undermine the accuracy and effectiveness of traditional design approaches. In this thesis, we formulate the problem of uncertainty aware mapping for multicore embedded system platforms as a multi-objective optimization problem. We present a solution to this problem that integrates uncertainty models as a new design methodology constructed with Monte Carlo and evolutionary algorithms. The solution is uncertainty aware because it is able to model uncertainties in design parameters and to identify robust design points that limit the influence of these uncertainties onto the objective functions. The proposed design methodology is implemented as a tool that can generate the robust Pareto frontier in the objective space formed by reliability, performance, and energy consumption.

Keyword: Embedded systems; Uncertainties; Robust mapping; Reliability; Performance; Energy consumption.

ACKNOWLEDGEMENTS

The last two years of my Master study life leave me indelible memories. With ups and downs, all the past days are still as fresh as yesterday. As the proverb says: "Feel appreciated when life does not forget". I feel so grateful for a lot of bright people practically or mentally supported and helped me during my Master study. Here, I would like to express my thanks to all these people.

First, I would like to thank my advisor: Dr. Cristinel Ababei. I feel fortunate that I have Cris to supervise my Master study! Cris provided me with an excellent working environment, which is full of trust and freedom. With this environment, I was able to pursue and explore my interested research areas and come up with my own ideas. In addition, Cris was always willing to make time for me for problem discussion, weekly meeting, paper revision, and so on. With his guidance, I was able to gain the ability of problem formulation and solution in doing research. What is more, apart from providing excellent scientific support and advice, Cris also taught me as to have a warm family and enjoy the happiness of the life. I can never thank him enough for his selfless dedication and guidance to me.

I would like to thank Dr. Richard Povinelli, who laid the foundation for my understanding of evolutionary algorithms in the Evolutionary Computing Course. Dr. Povinelli provided me many useful suggestions for improving my writing skills. Thanks for serving as one of my committee members and taking time to review my thesis.

I would also like to thank Dr. Henry Medeiros, who laid the foundation for my knowledge of advanced algorithms during the Algorithm Course. Dr. Medeiros shared his research experience with me: "Research needs focus". Many times I saw Dr. Medeiros focused on his research at late night in his lab. His research experience helped me leaped over the obstacles in front of my research areas fast and efficient. Thanks for being one of my committee members and spending time on reviewing my thesis.

An excellent thanks go out to Milad Ghorbani Moghaddam. From the start of my Master study, I spent most of my time working together with Milad in the MESS Lab. Thanks for helping me all the time! Furthermore, I would like to thank all the other colleagues in our lab: Ian Barge, Kellen Carey, Nate Zimmer, Shaun Duerr, Brandon Kupczyk, Jarrett Smalley, and Masoud Ghorbani Moghaddam. Thanks for all of you!

Meanwhile, I want to appreciate the friendship with many great Chinese and International friends during my stay in the USA. Thanks for helping me and making my life enjoyable in the past two years. They are: Jiangbiao He, Chen Li, Xiangyu Zhou, Tianyu Liu, Yun Bai, Jiao Qiao, Hao Chen, Yue Sun, Jiayi Su, Yuqin Wong, Dan Tong, Ann Li, Greg Merkel, Elizaveta Grushnikova, Reza Mdh, Yevgeniy Reznichenko, David Kaftan, Samuel Amoako-Frimpong, Philipe Dias and so on.

Finally, I would like to thank my parents and my cousin Weihua Guan, who supported my decision to study overseas. I owe my most profound gratitude to my wife, Qin Zhang, who supported and stood behind me all the time, through every hill, every valley. Without her, this thesis would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Problem Statement, Objective and Contributions

1.1 Problem Statement

Today, embedded systems can be found in many application domains, ranging from safety and mission-critical systems in avionics, automotive, nuclear plant control, and medical devices for multimedia, gaming, and communications. Future embedded systems will be increasingly complex and will contain tens to hundreds of heterogeneous cores. Due to continuous technology downscaling of fabrication processes, the design of embedded systems will face new challenges including: (1) increased design uncertainties due to variations in fabrication processes, supply voltages, and temperatures [1; 2]; (2) poor reliability and performance degradation caused by elevated rates of faults and increasingly adverse aging mechanisms [3; 4]; and (3) increased design complexity caused by heterogeneity of the hardware platform, diversity in hardware and software components, and new communication infrastructures such as networks-on-chip [5; 6].

In this thesis, we assume increased design uncertainties due to variations in fabrication processes, supply voltages, and temperatures, which have been discussed and modeled in recent literature [7; 8; 9]. Factors like these make for various design parameters or variables not to be deterministic anymore; instead, they become less precisely known or more uncertain, and many researchers started

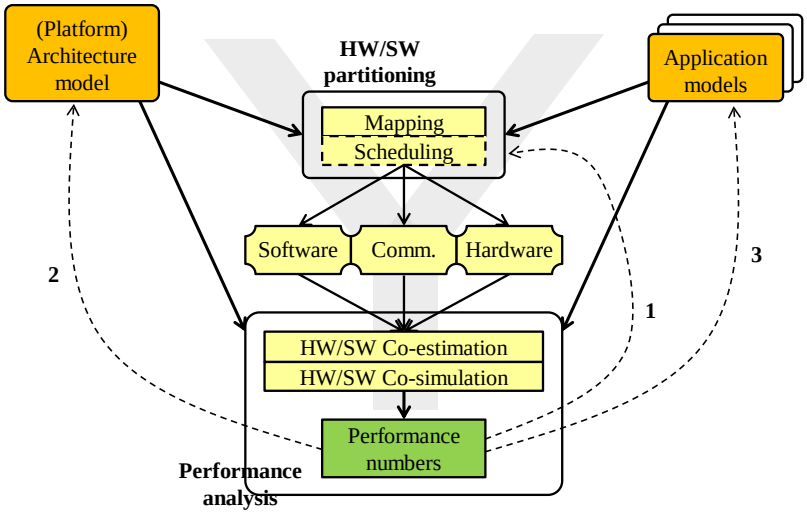
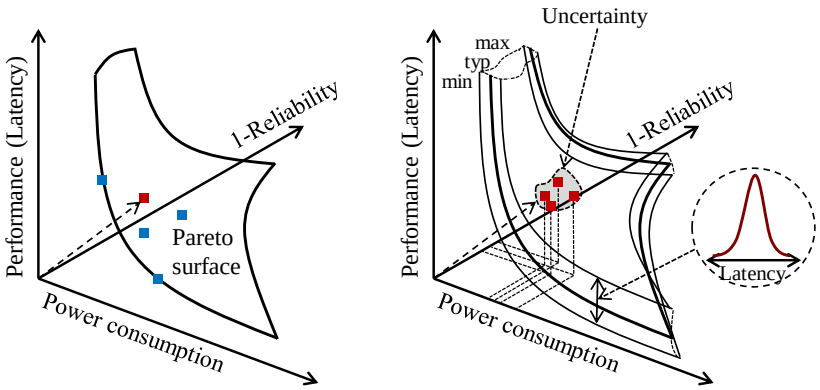


Figure 1.1: Conventional design flow for embedded systems. In this thesis, we focus on the problem of mapping. Dashed arrows labeled 1,2,3 indicate possible routes to go back in the design flow to change design decisions in order to improve the design.

to model them statistically rather than as fixed deterministic values. This uncertainty increases as we go to deeper nanometer technology nodes.

The traditional design process of embedded systems involves an automated design space exploration (DSE). DSE is an iterative process built mainly around the problem of mapping and scheduling of the application onto the architecture platform. The process typically follows a "Y-chart" design flow as illustrated in Fig. 1.1. During this DSE, the solutions generated and evaluated are as good and accurate (i.e., close to what they would be in reality in terms of different attributes such as reliability, execution time, and power consumption) as the accuracy of the model-based estimations that are employed. These estimations, in turn, rely on the accuracy of parameters that are used in the estimation models. If these design parameters become uncertain − increasingly so due to the reasons listed earlier − then, the optimization path during the design space exploration may become



(a) (b)

Figure 1.2: (a) Pareto frontier surface in traditional embedded systems design. (b) Uncertain Pareto surface where a design point degenerates into multiple solutions.

uncertain and diverge from the path towards the true optimal design solution.

In this context, it becomes desirable to be able to quantify such divergence and to develop a design methodology capable of finding design solutions that are the most likely, with a certain confidence, to be robust against uncertainties. These design solutions represent points on the Pareto frontier generated during the design space exploration, an example of which is shown in Fig. 1.2.a. However, when one considers uncertainties in the design process, the traditional Pareto surface in the solution space becomes uncertain as shown in Fig. 1.2.b. *This is the problem addressed in this thesis. As it will be described next, we propose a solution to the problem of mapping under uncertainties.*

1.2 Objectives

In this thesis, we propose a design method that is able to identify robust design points on the uncertain Pareto frontier. The proposed method models and handles uncertainties directly. This method is implemented as a computer program (i.e., a

design tool) that integrates uncertainty models and algorithms to solve the problem of mapping for hardware/software (HW/SW) design of embedded systems. These algorithms are capable of performing robust multi-objective optimization to effectively balance reliability, performance, and energy consumption. This tool will help embedded systems designers to identify the best design solution points on the uncertain surface from Fig. 1.2.b under assumed levels of uncertainties. This tool chooses as the best final solution the one closest to the "origin" of the 3D objective space from Fig. 1.2.b. The chosen solution represents a compromise among all three objectives. However, the designer can pick a different solution. For example, if performance is the most important for some application, then, a design point with the best performance can be selected, but likely with worse reliability and power consumption.

1.3 Contributions

This thesis proposes a solution to the problem of mapping for embedded systems under uncertainties. To this end, the main contributions of this thesis include:

- A solution to the mapping problem for general purpose embedded systems while considering simultaneously reliability, execution time, and energy consumption. The solution is implemented as a design space exploration framework tool called DESUU (Design of Embedded Systems Under Uncertainty), which uses the Non-dominated Sorting Genetic Algorithm (NSGA-II).

- Models of uncertainty in design parameters. We investigate different levels of injected uncertainty and provide simulation results.

- A novel uncertainty aware analysis technique with consideration of both uncertainty correlations and different levels of uncertainty. The proposed uncertainty aware analysis technique is implemented as a framework tool called DESUU-II.

- Simulation results that demonstrate the advantages of the proposed techniques and solutions. In this thesis, we analyze an architecture platform constructed with both hardware and software components.

To the best of our knowledge, this work is the first to address the problem of multi-objective (reliability, performance, and energy) mapping for general purpose embedded systems under uncertainties.

1.4 Thesis Organization

The organization of this thesis is depicted in Fig. 1.3. Chapter 2 provides background information. It starts with the overview of Electronic System-Level (ESL) design and the principle of Design Space Exploration (DSE). Then, it continues with a more detailed discussion of recent work on uncertainty aware and reliability oriented embedded systems design. Chapter 3 presents a description of the Anti-lock Brake System (ABS) application testcase, which is used in the motivating discussion for the proposed method. In Chapter 4, we present the uncertainty models. Then, we present the proposed method for solving the problem of mapping in embedded systems design under uncertainty. Chapter 5 presents the comparison simulation experiments with the traditional method. Chapter 6 presents the simulation experiments of the proposed robust method, as well as a computational

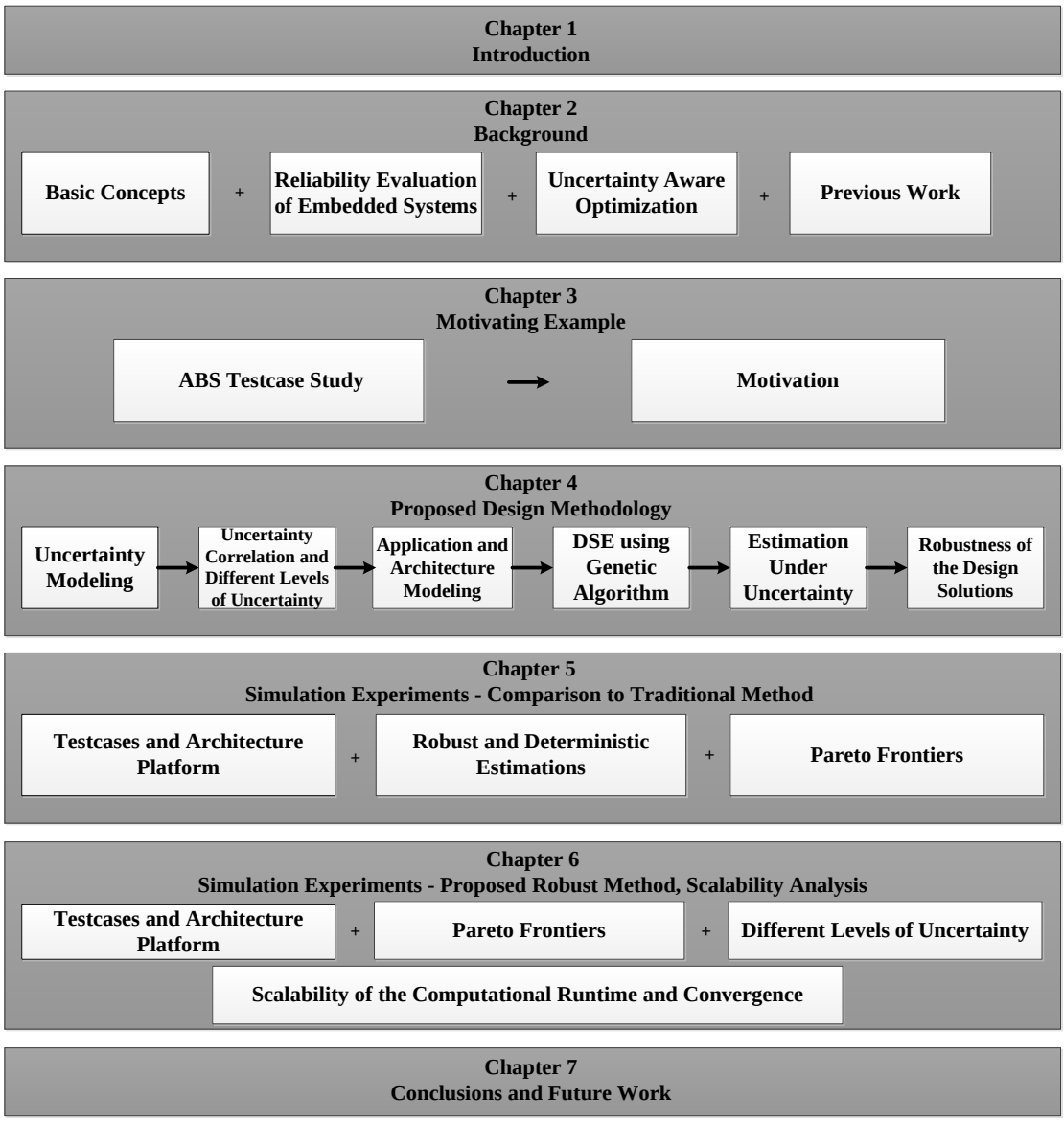


Figure 1.3: Thesis outline.

complexity analysis to study the scalability of the implemented tool. Finally, Chapter 7 concludes the thesis and discusses future work ideas.

CHAPTER 2

Background

In this chapter, we first introduce an overview of Electronic System-Level (ESL) design, which points out the level of abstraction for our target problem. Then, we introduce the Design Space Exploration (DSE) and the main challenges in this domain. In the third section, we review traditional classes of widely-used reliability evaluation approaches for embedded systems. In the fourth section, we introduce the basic techniques that are used for uncertainty aware optimization of embedded systems. Finally, we review the previous work related to the problem of mapping of embedded systems with the design objectives that include reliability, performance, and energy consumption.

2.1 Electronic System-Level Design

The term hardware/software codesign appeared in the early 1990s to describe the confluence of hardware and software in Integrated Circuit (IC) design [10; 11]. Due to technology downscaling, the ICs have increased in complexity, time-to-market pressure, and development costs, the abstraction level at which the systems under design are expressed must be solved [12; 13]. The term Electronic System-Level (ESL) design, at which interfacing and reusing designs across different abstraction levels are facilitated, is resulted by these challenges. Hardware/software codesign at ESL reduces the time-to-market and design risks through the simultaneous

analysis, exploration, and design of hardware and software [14]. Nowadays, the major challenges in the design of electronic systems as stated in [15; 16] are:

- *Allocation or architecture synthesis:* It is the process of selecting a set of system resources such as processors, hardware intellectual property (IP) blocks, and their interconnects that compose the system architecture.
- *Mapping:* It is the process of mapping system functionality using tasks, processes, functions, and so on, onto the system architecture.
- *Scheduling:* It is the process of ordering the execution of functions, memory accesses, and communications on individual resources.

The set of all permutations of allocations, mapping, and scheduling decisions determines the design space of embedded systems.

2.2 Design Space Exploration

The task of system synthesis is defined as the allocation of resources from the architectural model, mapping of the tasks onto the allocated resources, and scheduling the execution order of the tasks [17]. The feasible design solutions are represented by the permutations of allocations, mappings, and scheduling decisions that satisfy the given design constraints. The process of finding these feasible design solutions is called Design Space Exploration (DSE). Regarding a large number of design alternatives, such as the type and the number of processors, memory units, and interconnections, the design space is usually extremely huge, prohibiting manual search. In addition, the design objectives, such as reliability, performance, and

energy consumption, of DSE are usually complex and related to each other, increasing the DSE complexity. Therefore, it is important to have a systematic DSE that is automated as much as possible at the early stage of embedded system design. Such automatic DSE is an iterative process built mainly around the problem of mapping and scheduling of the application tasks onto the architecture platform. For instance, Fig. 1.1 in Chapter 1 illustrates the "Y-chart" design flow. The main challenges in the DSE domain are:

- *Exploration techniques:* DSE requires good exploration algorithms which are suitable for large discrete search spaces with a large number of alternative solutions and multiple design objectives. For instance, in this thesis, we investigate the performance of an evolutionary algorithm to find design solutions for DSE with multiple design objectives.

- *Evaluation of design solution points:* How to model and evaluate the flexibly and efficiency of different design objectives is another challenging problem. For example, in this thesis, we customize the evaluation function with a Monte Carlo simulation technique, to evaluate the design solution points with injected levels of uncertainty in design parameters.

2.3 Reliability Evaluation of Embedded Systems

Reliability has been a primary design objective of DSE of embedded systems. Research into reliability evaluation of architecture platform has led to a variety of models, each of which focuses on a specific level of abstraction or system characteristics. This section reviews essential classes of widely-used and well-accepted

reliability evaluation models for embedded systems.

2.3.1 Combinatorial Models

Combinatorial models usually decompose the complex system into functional entities, such as units or subsystems, for reliability evaluation. Some of the classic combinatorial models are Reliability Block Diagram (RBD) and Fault Trees (FT).

Reliability Block Diagram

RBD is a reliability modeling approach in reliability evaluation of embedded systems architecture platform. An RBD models the structural relationship of how the sub-system failure and the components failure combine to lead to system failure. When a RBD approach is used, the system is decomposed into *Reliability Blocks* that have particular failure characteristics. The connections between the reliability blocks construct the path of the system behavior. If it is possible to find at least one way from the start of the RBD to a particular component through operational components, the particular component is considered functional. These components can be organized in series, parallel or other structure. An organization of a set of blocks of components that are configured in parallel within the blocks is called Series-Parallel (SP) system [18]. In SP systems, we call *subsystem* for a component with its parallel redundancies, and we compute the reliability of the subsystems independently from the other parts of the system. Therefore, in SP RBDs, the overall system becomes a series of connected subsystems whose reliability is known. Hence, the overall reliability of the system can be analytically

computed under SP assumption. RBD is a reliability modeling approach in reliability evaluation of embedded systems architecture platform. An RBD models the structural relationship of how the sub-system failure and the components failure combine to lead to system failure. When a RBD approach is used, the system is decomposed into *Reliability Blocks* that have particular failure characteristics. The connections between the reliability blocks construct the path of the system behavior. If it is possible to find at least one way from the start of the RBD to a particular component through operational components, the particular component is considered functional. These components can be organized in series, parallel or other structure. An organization of a set of blocks of components that are configured in parallel within the blocks is called Series-Parallel (SP) system [18]. In SP systems, we call *subsystem* for a component with its parallel redundancies, and we compute the reliability of the subsystems independently from the other parts of the system. Therefore, in SP RBDs, the overall system becomes a series of connected subsystems whose reliability is known. Hence, the overall reliability of the system can be analytically computed under SP assumption.

Fault Trees

FT is another reliability evaluation approach that is widely used in the literature [19; 20]. FT construction is a deductive, top-down process where the failure events are organized into a tree structure. In the analysis of the reliability of the target architecture platform, the effects of lower-level faults and events are systematically propagated by quantifying the reliability of a higher-level abstraction. Different types of FTs, such as Component Fault Trees (CFT) [21] and State Event Fault

Trees (SEFT) [22], are used in the reliability evaluation in different contexts and abstraction levels.

2.3.2 Markov Model

Markov modeling is used for analyzing complex probabilistic systems taking into consideration of repair mechanisms and the order of events in the system. A Markov model is constructed by a set of equations that describe the probabilistic transitions among the states and initialization probability distributions of the starting states. One important property of a Markov model is that the current state transitions are independent of the history of the state transitions, *i.e.* transition from state i to state j depends only on the state i, and is independent of the history that led to state i. This property indicates that the complete history in Markov model is summarized in the current state of the process.

Discrete Time Markov Chains (DTMC)

DTMCs are finite state machine formalisms with probabilities of transitions between states that are widely used in modeling discrete-time dynamic systems. A DTMC can be applied to represent all the relevant states of software execution and the probability to transfer from one state to another. Among all the states, the starting state is called the *initial state* and one or more among the other states represent successful completion of execution or occurrence of a failure. The formal definition of a DTMC can be expressed as a tuple (S, s_0, P, L) where,

- S is a finite set of states

- s_0 is the *initial state*
- $P : S \times S \rightarrow [0, 1]$ is the *transition probability matrix*
- $L : S \rightarrow 2^{AP}$ is the *labeling function*

DTMCs can be used to model both a single transition system and the synchronous composition of many systems. The labeling function describes the mapping process from the states to the set of atomic propositions (AP). $P(s, s')$ denotes the probability of making a transition from state s to the states s'. In a DTMC, $\sum_{s' \in S} P(s, s') = 1$ for all state $s \in S$, which implies that even terminating states should have an outgoing transition to itself with a probability 1. When a system is modeled as a DTMC, the execution is represented by a path through the DTMC.

2.4 Uncertainty Aware Optimization of Embedded Systems

Uncertainty aware optimization methods for embedded systems have been proposed only recently. The work in [23; 24] are the latest attempts to address the problem of uncertainty in reliability evaluation. Both [23; 24] use a Monte Carlo Simulation techniques to handle uncertainty in design parameters, and evolutionary algorithms for optimization of reliability.

2.4.1 Monte Carlo (MC) Simulation

The MC simulation takes samples from the input parameters of the architectural elements, which vary in the probability distribution. Any sampled parameter of

an architectural element may be contributed to more than one parameter in the evaluation model. Every time a sample is taken from an input distribution, all model parameters dependent on this parameter are updated. The steps that are involved in the MC simulation are as follows:

- *Sample:* A sample is taken from the probability distributions of each parameter. We draw a sample from these distributions as follows:

  (a) Obtain the Cumulative Distribution Function (CDF) of the parameter from its PDF.

  (b) Generate a random number x from the uniform distribution $(0, 1)$.

  (c) Obtain $CDF^{-1}(x)$.

- *Update:* From the samples obtained from the input distributions, the numerical values for the evaluation model parameters are updated. Since more than one parameter of the probabilistic model may refer to a setting in the architecture, a subscription mechanism is proposed. Parameters of the evaluation model are subscribed to uncertain parameters in the architecture platform. When we sample a parameter from the input distribution for a specific architectural setting, all the subscribing model parameters are updated and recomputed.

- *Compute:* Analytically simulate the model and obtain the computed results.

2.4.2 Evolutionary Algorithms for Optimization

For the design of embedded systems with multiple conflicting design objectives, one can provide either a weight function to combine multiple objectives or a multi-objective exploration technique to scan the search space simultaneously. The former method needs to choose the proper weight coefficients for the optimization function and results in a single optimized solution. In contrast, the later method determines not only one optimized solution but rather a set of Pareto frontiers. Therefore, population-based methods, such as evolutionary algorithms, have received a lot of attention in this area. Among them, the Non-dominated Sorting Genetic Algorithm (NSGA)-II has been shown to perform efficiently for system-level synthesis.

For the optimization process, NSGA-II uses an initial population of chromosomes consisting of alleles. Each allele in a chromosome represents a mapping of a task from the application to a component in the architecture platform. The initial population of deployment architectures is generated at random. The crossover and mutation operators are used to create new chromosomes by combining existing ones or changing the mapping of a single task to another component on the architecture platform. Then, all the chromosomes are evaluated, according to the evaluation function in NSGA-II, and selected to form the new parents' population. Details about the NSGA-II will be discussed in Chapter 4.

2.5 Previous Work

In this thesis, we focus on the problem of mapping of embedded applications to multicore systems-on-chip (SoCs) platforms with consideration of specified levels of uncertainty and with the primary objectives that include reliability, performance, and energy consumption. The problem of HW/SW co-design for embedded systems has been studied extensively in the past.

It was formulated as multi-objective optimization in studies of *system-level synthesis* [25; 26; 27; 28] as well as of *platform configuration* [29]. The former focuses on solving the problem of mapping a task-level application onto a heterogeneous architecture constructed with both hardware and software components. The latter includes parameter tuning for the platform architecture and its configuration space exploration. The work in [25] solves this multi-objective optimization problem by using multi-objective evolutionary algorithms (MOEAs). Simulation results showed that MOEAs provide the designer with a set of solutions in a reasonable amount of time. The authors of [27] apply a *divide-and-conquer* approach to solve the multi-objective mapping problem. The study in [26] focuses on evaluating the performance of various state-of-the-art task mapping heuristics, both at design time and at runtime, by using the rSesame framework on a reconfigurable architecture. The work in [28] proposes a hybrid task mapping algorithm, which combines a static mapping exploration and a dynamic mapping optimization for heterogeneous MPSoCs, and achieves an overall improvement of system efficiency. However, that work only considers performance as the main design objective. The study in [29] focuses on exploring architectural parameters, such as processor type,

memory subsystem, and bus communication, that make up the hardware kernel of a parameterized SoC platform for the design of embedded systems with the consideration of power consumption and performance constraints.

Several previous solutions to the mapping problem have been integrated into computer-aided design (CAD) automation tools. For instance, these tools include architectural exploration environments, such as those described in the following paragraphs:

Metropolis [30] is an integrated electronic system design environment for simulation, formal analysis, and synthesis of embedded systems. It is based on the *metamodel* concept, which can support not only functional capture and analysis but also architecture description and the mapping of functionality to architectural elements. The Metropolis metamodel's formal semantics allow embedding computation models into a rigorous framework that favors design reuse and design chain support. It was used for applications from automotive to wireless communication and video applications.

MESH [31] is a performance modeling environment which captures software-on-hardware in concurrent, layered thread relationships in SoC designs. It provides a primary interface between functional and *instruction set simulator* (ISS) models and allows for early and high-level performance modeling without the need for the knowledge of ISS or complete software models. It also efficiently tracks heterogeneous design trade-offs while considering design objectives that include performance. However, as an interface between the high-level functional model and the low-level ISS model, it has an increased development complexity.

SCE [32] is a system-level design framework, which uses the SpecC specification language. It follows a *specify-explore-refine* methodology, with support for heterogeneous platforms constructed of both hardware and software components, IP blocks, and buses for communication. It is an automated design flow with a toolchain from specification down to hardware/software implementation. It allows rapid and extensive design space exploration and thus can find out an optimal implementation quickly.

Artemix [33] is a workbench that provides modeling and simulation methods and tools for evaluating performance efficiently and for exploring design space of heterogeneous embedded multimedia systems. By transforming dataflow actors in the intermediate mapping layer, and transforming coarse-grained application events into finer grained architecture events, it can bridge the abstraction gap between application and architecture models. It is composed of mainly two system-level modeling and simulation environments, which make it powerful but complicated.

ESPAM [34] aims at automating multiprocessor system design, programming, and implementation. It transfers the design specification and programming from the Register Transfer Level (RTL) and C level to a higher level of abstraction of the system level. When it is applied, it first specifies a multiprocessor system at a high abstraction level. Then, it refines this specification down to a real implementation. ESPAM reduces the design time beginning from the system-level specification and going down to complete implementation.

SHARA [35] is a scenario-based hierarchical run-time adaptive resource

allocation framework. This framework integrates a hierarchical resource management mechanism, where a global resource manager controls the workload distribution among tiles and the local resource manager optimizes the resource allocation for the assigned applications to reduce the complexity of the task mapping problem at runtime. It also includes a hybrid approach, which combines the design-time optimization of DSE with run-time mapping re-optimization, for mapping applications to the underlying resources, and thus handle the complex and dynamic application workloads for MPSoC systems. In addition, SHARA includes a self-adaptive scheduler for adaptivity throttling. SHARA can support large numbers of workload scenarios with near-optimal mappings. It can also adapt its behavior according to the user behavior. However, it does not consider the power consumption and reliability or uncertainty as design objectives.

These tools facilitate flexible system-level performance evaluation by providing support for mapping a behavioral application specification to an architecture specification. *However, most of these tools have not considered reliability or uncertainty.* Reliability has become a primary design concern in optimization techniques of embedded systems. The review in [36] discussed several studies that focused on architectures constructed only with software components. The authors pointed out that, at that time, only a few researchers directly considered uncertainty and/or reliability as design objectives. The study in [37] formulates a framework to evaluate the system reliability under uncertainty. The work in [38] introduces a simulation-based method which uses Discrete Time Markov Chains (DTMC) and probabilistic model checking to accommodate a diverse set of parameter range distribution when measuring the uncertainty. The work in [39] proposes

to automatically incorporate Imperfect Fault Coverage (IFC) into the reliability model, in order to accurately analyze the reliability of complex systems including nested redundancies and repeated components. Their approach can evaluate system reliability more accurately at reasonable computation time and memory overhead compared to previous IFC-aware approaches.

Reliability has become a primary design concern also in networks-on-chip (NoC) and multicore processors as well. As such, it started to be considered alongside more traditional design objectives like performance and energy consumption. For example, the study in [40] presented a run-time resource manager that finds the most effective mapping of tasks on the processing nodes to optimize system reliability while leveraging on performance and communication energy in NoC-based many-core architectures. Similarly, the study in [41] presented a neural network based reliability estimator and thread migration for dynamic reliability management for chip multiprocessors. Furthermore, the work in [42] investigates the use of dynamic voltage and frequency scaling (DVFS) as a mechanism for dynamic reliability management for chip multiprocessors.

However, the majority of the previous work did not consider uncertainty or reliability in the design process of embedded systems. The studies in [23; 24] are recent attempts to capture uncertainty in the process of optimization of embedded systems. The work in [23] proposed a novel robust optimization approach that deals with uncertain parameters during the design phase of software-intense systems. But, reliability was the only objective considered during the optimization process. In addition, the authors only focused on architecture platforms constructed with

software components. The study in [24] proposed an uncertainty-aware reliability model for the design space exploration of embedded systems. But, similarly to the study in [23], reliability was considered as the only uncertain parameter. In addition, the authors made an unstated assumption that the components are affected by one uncertainty source (e.g., one correlation group). In fact, the components are usually affected by multiple uncertainty sources with different levels of uncertainty, and the uncertainty in design parameters is the combination of the influence of different uncertainty sources. For instance, the failure rates of the components of an embedded system from the automotive application domainshich are located close to both the engine and the cooling fan can be affected by both the engine heat and the cool air from the fan. Moreover, the authors did not explore the impact of different levels of uncertainty on the design parameters.

Therefore, while these works focused on formulating reliability estimation techniques with consideration of uncertainties, in this thesis, we take that further and integrate such techniques in a more comprehensive approach that also considers performance and energy consumption, not only reliability. In addition, we investigate different levels of uncertainties and analyze both uncertainty correlation and different levels of uncertainty.

2.6 Summary

This chapter presented the background and the related work for uncertainty aware mapping of embedded systems for reliability, performance, and energy consumption. In next chapter, we will introduce the motivation of this thesis.

CHAPTER 3

Motivating Example

In this chapter, we introduce the Anti-lock Brake System (ABS) application testcase. The ABS testcase study represents a specific problem from the automotive industry. It maps the components of the ABS application to the heterogeneous architecture of the embedded system. We use it to present the motivation for the work proposed in this thesis. We use both the traditional point-estimate and the proposed robust-estimate approaches to evaluate the system reliability, execution time, and power consumption of the ABS testcase. Simulation results show that there is a significant difference between these two estimation approaches. Therefore, we conclude that a new design method, which is capable of modeling uncertainty to provide reliable and robust design solutions, is needed for the design of future high performance heterogeneous embedded systems.

3.1 ABS Testcase Study

The ABS is designed to optimize the braking effectiveness in order to keep wheels rolling on the road and to reduce the breaking distance [43; 44]. It is important for the car control as it is used to prevent the lockup of the wheels during the braking action [45]. The block diagram of a typical ABS is shown in Fig. 3.1. It includes the blocks labeled from 0 to 7, as components that interact as shown by the arrows. The block labeled 0 is called the ABS main unit. Its role is to

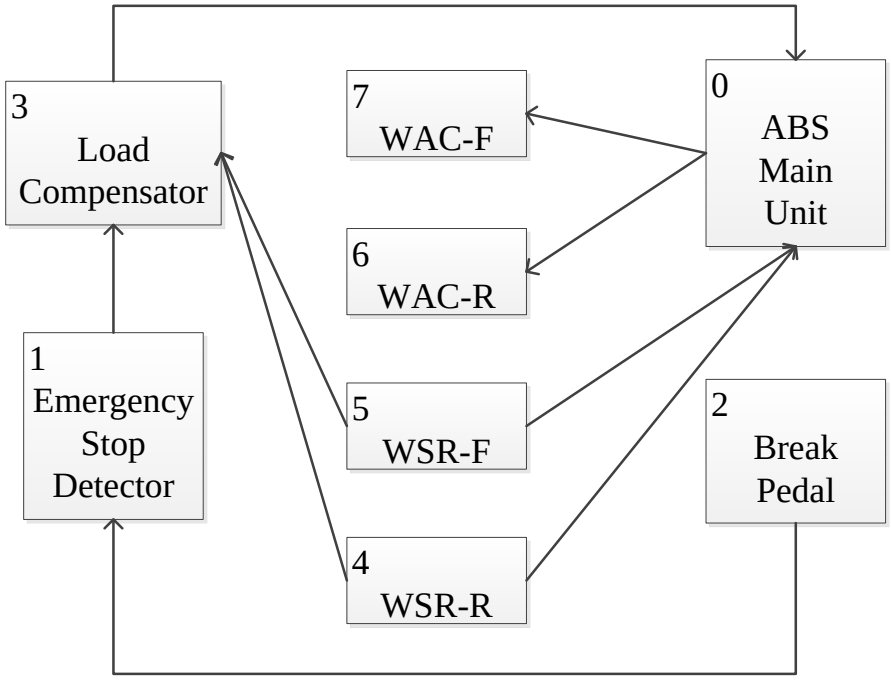


Figure 3.1: ABS software components and interactions.

prevent skidding and to help drivers control the wheels on wet and slippery roads. Block 1 is the Emergency Stop Detector, whose role is to maximize the brake pressure if it detects any sudden pedal action associated with an emergency stop. Block 2 is the Brake Pedal Sensor, whose role is to read from the pedal sensor and send the data through to Block 1. Block 3 is the Load Compensator, which is used to improve the braking performance by compensating uneven braking due to the heavy or unbalanced loading of the vehicle [46]. Blocks 4 to 7 represent the transceiver software components dedicated to each wheel, which communicate with sensors and brake actuators. More specific, WAC stands for wheel actuator controllers and WSR represents wheel sensor readers.

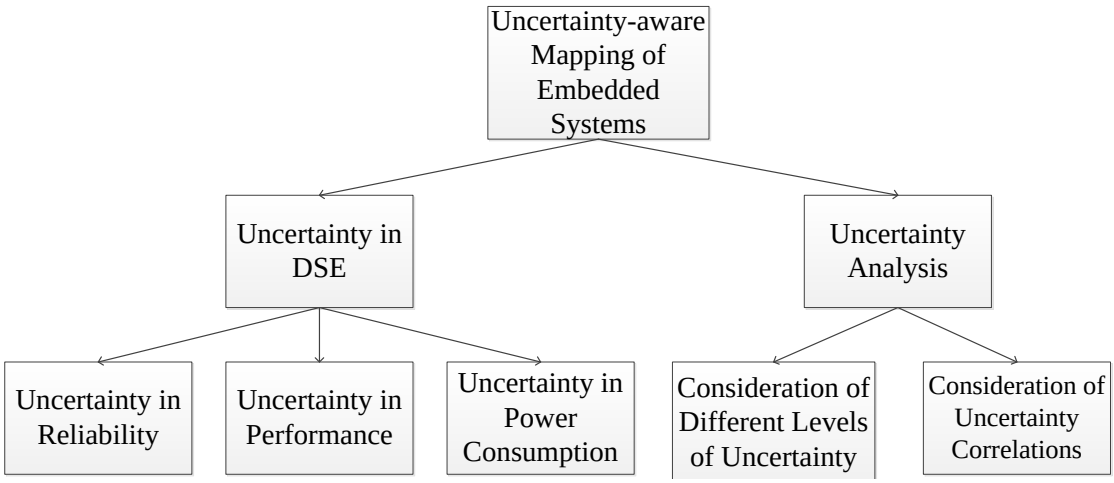


Figure 3.2: Motivation of this thesis.

3.2 Motivation

If we used traditional design methods to map the ABS testcase described above, then, in such methods we would work with point-estimate parameters. However, these parameters can become uncertain as discussed in Chapter 1. If design parameters become uncertain (e.g., the failure rate of CPUs who are located close to the engine or the cooling fan can be significantly affected by the different temperatures), then, the design space exploration to solve the mapping problem may lead to suboptimal solutions. In building the motivation for this thesis, we ask ourselves two questions, as illustrated hierarchically in Fig. 3.2:

1. Why do we need to consider uncertainty in design parameters of embedded systems? In addition, why do we need to consider uncertainty in the estimation of reliability, performance, and energy consumption?

2. Why do we need to consider both different levels of uncertainty and uncertainty correlations?

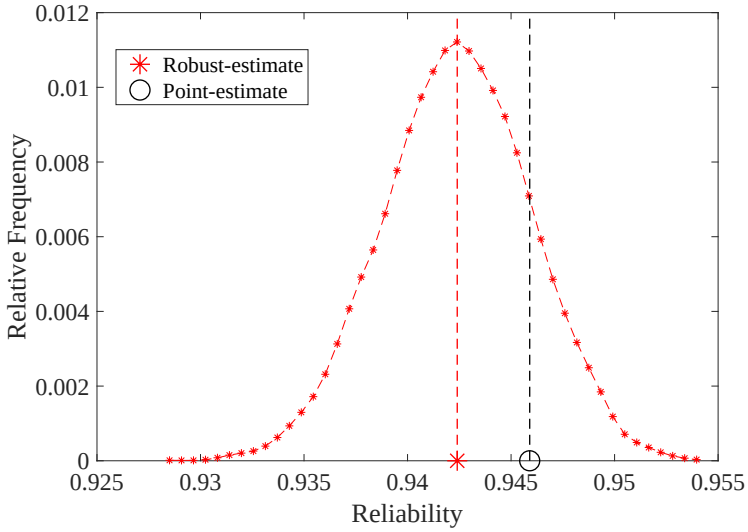


Figure 3.3: Histogram of system reliability. All histograms are obtained using 10^5 parameter samples during the estimation process.

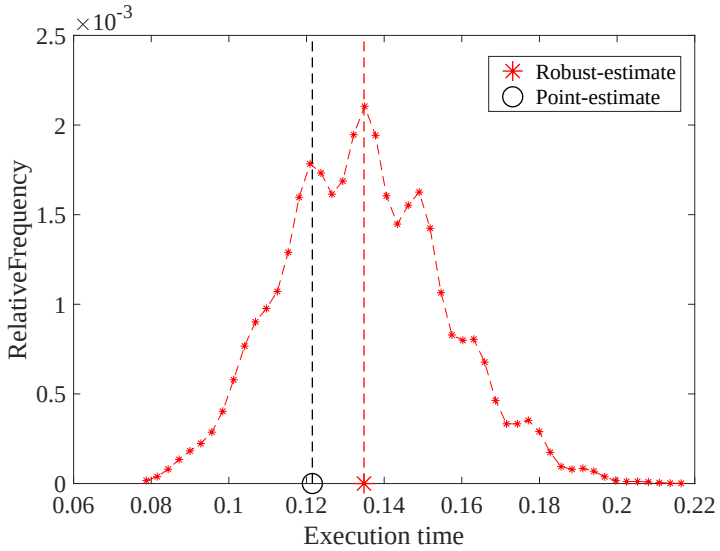


Figure 3.4: Histogram of execution time. All histograms are obtained using 10^5 parameter samples during the estimation process.

3.2.1 Answering Question 1

In answering the first question, we used the proposed mapping algorithm described later in this thesis to map the ABS testcase to a platform architecture also described later in Chapter 6. The proposed method has the ability to model uncertainty in design parameters. We use it to estimate the reliability, which we consider

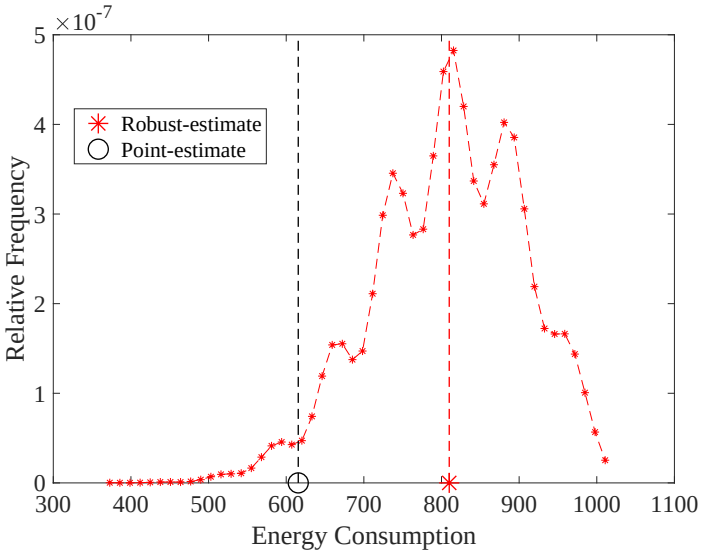


Figure 3.5: Histogram of energy consumption. All histograms are obtained using 10^5 parameter samples during the estimation process.

to be affected by uncertainty. Details about the estimations under uncertainty for reliability, performance, and power consumption will be described later in Chapter 4.

In order to report the difference obtained when considering uncertainty in design parameters of embedded systems, we implement both the robust-estimate approach and the point-estimate approach in our framework which will be discussed later in Chapter 5. In this experiment, we use the same given mapping (e.g., round-robin mapping) when estimating the design objectives such as system reliability, execution time, and energy consumption through both approaches. The reliability, execution time, and energy consumption of the system as estimated by our tool are shown in Fig. 3.3, Fig. 3.4, Fig. 3.5. This figure also shows the estimated reliability, execution time, and energy consumption when design parameters are estimated using the traditional point-estimate approach. Fig. 3.3 depicts the histogram of reliability obtained using 10^5 samples during the Monte

Carlo estimation. It can be seen that the mean value of the estimated reliability is 0.9424, while the point-estimate reliability is 0.9459. This represents a 13.4% difference between these two estimations. From this experiment, it can be concluded that when design parameters are subject to uncertainty, estimation of reliability with the traditional point-estimate approach may be significantly inaccurate. In other words, if we use traditional point-estimate values for reliability, we would overestimate it. This may lead to suboptimal solutions.

Similarly, if we consider uncertainty in design parameters that affect performance and energy consumption, the proposed tool provides the estimations shown in Fig. 3.4 and Fig. 3.5 Again, we can see significant differences between these estimations and the estimated values with the traditional approach. Fig. 3.4 shows that the mean value of the estimated execution time is $0.1348s$, while the execution time of the point-estimate approach is $0.1215s$. This represents a 9.5% difference. Similar results are found for the estimation of energy consumption. The energy consumption estimated with the proposed tool is $809.9774uJ$, while the point-estimate is $615.51uJ$. This represents a 29.9% difference between these two estimation approaches. Therefore, in answering the question why to consider uncertainty in design parameters, we find out that there are significant differences between the traditional point-estimate and the robust-estimate of reliability, execution time, and energy consumption estimations. Large differences in these estimations may lead to suboptimal design solutions when design parameters are affected by uncertainties.

To address that, the next chapter proposes a design method capable of

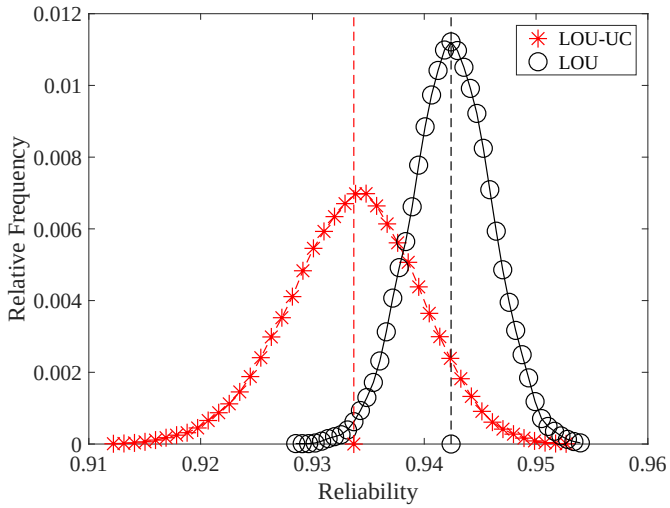


Figure 3.6: Histogram of system reliability for LOU and LOU-UC techniques.

robust multi-objective optimization. This method is implemented as a computer-aided design (CAD) automation tool constructed with Monte Carlo and evolutionary algorithms, which can overcome the issues described earlier.

3.2.2 Answering Question 2

Existing uncertainty-aware analysis techniques consider either only Uncertainty Correlations (UC) or different Levels of Uncertainty (LOU). However, in practice, it is possible that factors like temperature may affect several system components simultaneously. This introduces uncertainty correlations between system components. In addition, these components are usually affected by multiple uncertainty sources with different levels of uncertainty. Thus, different levels of uncertainty exist between these system components. Therefore, we need to consider different levels of uncertainty as well as uncertainty correlations when we analyze uncertainty in design parameters.

In order to report the difference obtained with and without the consideration of uncertainty correlations, we implement in the proposed tool the capacity to model both situations: different levels of uncertainty and uncertainty correlations. We use the same ABS testcase and the same mapping as in Section 3.2.1 to evaluate the design solution in both situations. Fig. 3.6 shows the histogram of reliability obtained with LOU (DESUU-I) and LOU-UC (DESUU-II) techniques using 10^5 samples during the Monte Carlo estimation. We can see that there exists a significant difference between the two mean values of the estimated reliability. This shows that when we consider only different levels of uncertainty, the estimated reliability spans a relatively small range with high probability. In the case of the LOU-UC, there exists a much smaller difference in the relative frequency value of bounds and of the mean value of the estimated reliability. This shows that the components affected by the same uncertainty sources, which is captured as uncertainty correlations, may result in a reliability distribution significantly different. In other words, if we consider only the different levels of uncertainty of the components in our modeling, we may overestimate the system reliability, which can lead to inaccurate reliability estimation.

To address this issue as well, in the next chapter, we propose a novel uncertainty-aware analysis technique to consider or capture both aspects: different levels of uncertainty and uncertainty correlations of the components.

CHAPTER 4

Proposed Design Methodology

How can we design high performance heterogeneous embedded systems that are reliable and robust to uncertainty in design parameters? This chapter seeks to answer this question by laying the foundation for uncertainty modeling and robust multi-objective optimization for embedded systems design. A design flow, which is capable of robust multi-objective optimization incorporated within a CAD automation tool constructed with Monte Carlo and evolutionary algorithm, is developed. The proposed design flow is uncertainty-aware in the sense that it is able to capture and directly deal with uncertainty in design parameters. It is reliability-oriented as reliability is included as a design concern in addition to performance and energy consumption. The proposed probabilistic uncertainty models and algorithmic innovations to solve the multi-objective mapping problem are discussed in the following sections.

4.1 Approach Overview

The proposed design flow is essentially an iterative process that uses an enhanced evolutionary algorithm, to solve the problem of mapping. The problem of mapping is the problem of finding the best placement of application tasks and communications between tasks onto the architecture platform.

The block diagram of the proposed design flow under uncertainties is

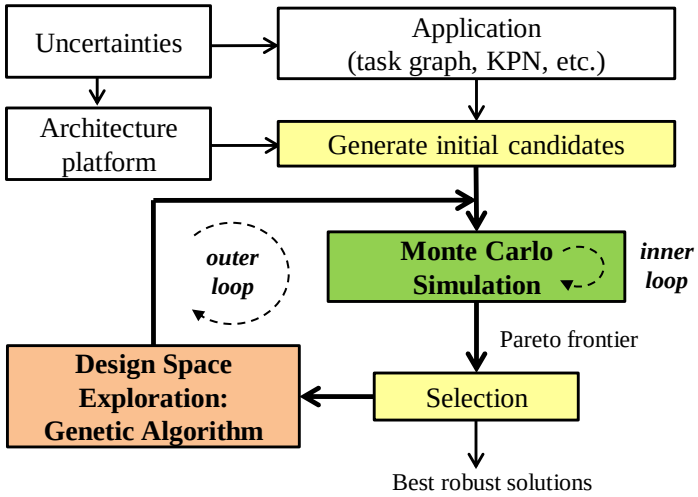


Figure 4.1: Block diagram of the proposed design method for embedded systems mapping under uncertainties.

shown in Fig. 4.1. The *outer loop* represents the iterative process of the design space exploration. The *inner loop* represents the iterative process of the Monte Carlo simulation technique that we employ for the estimation of objective functions under uncertainty. The primary objectives that we consider in this thesis include reliability, performance (measured as execution time), and energy consumption. Thus, the problem we attempt to solve is a multi-objective problem under specified levels of uncertainty. The output of the optimization process illustrated in Fig. 4.1 is a set of robust solution points that form the robust Pareto frontier in the three dimensional objective space *(1-reliability) vs. performance vs. energy consumption*. In the next sections, we describe the primary steps of the proposed design flow. These steps correspond to different blocks from the diagram in Fig. 4.1.

4.2 Uncertainty Modeling

The proposed design flow directly considers the uncertainty in design parameters and variables. The design parameters in embedded systems can be grouped into system-specific parameters (e.g., hardware and software failure rates, throughput metrics) and environment-related parameters (e.g., operational and usage profile of the system). The accuracy of these parameters is adversely affected by various kinds of uncertainties. Generally, it is difficult to determine accurate values of some of these parameters. Other parameters require information that is only available in later stages of the design process or depend on application-specific workloads, and thus are only available at runtime. Hence, the parameter values used in design-time optimizations represent estimations that are subject to uncertainty.

Uncertainty arises from the lack of knowledge regarding the true value of a quantity of interest. Uncertainty implies that optimization decisions might be non-optimal because one might expect one outcome but something quite different might in fact occur. Generally, uncertainty is heterogeneous and diverse. Sources of uncertainty include: uncertainty of data and model parameters, uncertainty about model choice, and uncertainty about the future. In the nanometer scale domain, uncertainty arises from temperature and voltage gradients, variations in application workloads, and from fabrication process and circuit parameter variations. Many design parameters and variables can be affected by uncertainties. For instance, failure rates of software components depend on the amount of testing and complexity of the algorithms contained in the component. Likewise, the failure rate of a hardware component can depend on the operational environment.

Hence, capturing parameter uncertainty into hardware platform and application description is difficult.

In the proposed design flow, we employ probability distributions to specify design parameters affected by uncertainty. This approach allows the design parameters to be given as probability distributions (continuous or discrete) in any mixture. The use of probability distributions entails the support for conversion from other complementary approaches. For example, interval estimation can be represented as a uniform distribution while the mean-variance estimation methods can be replaced with a normal distribution with the same mean and variance [23].

The *Uncertainties* block on the top left-hand side from the diagram in Fig. 4.1 represents the uncertainty injection process. There has been significant work studying uncertainty in various fields including engineering, mathematics, and other sciences [47; 48; 49]. However, it is generally agreed that there is no single model for handling any type of imperfect information. Therefore, similarly to [23], we propose to adopt the most general approach to capture uncertainty: design parameters and their variation can be specified as generalized, continuous or discrete, probability distributions in any mixture. Aside from its generality and ability to accommodate any probability distribution, this approach has the advantage of being able to accommodate complementary approaches as well. For instance, we can use uniform distributions to convert interval estimates into the proposed framework. On the limitations side, combining different probability distributions is usually analytically intractable, and therefore we must resort to Monte Carlo simulation based techniques in order to quantify figures of merit of interest

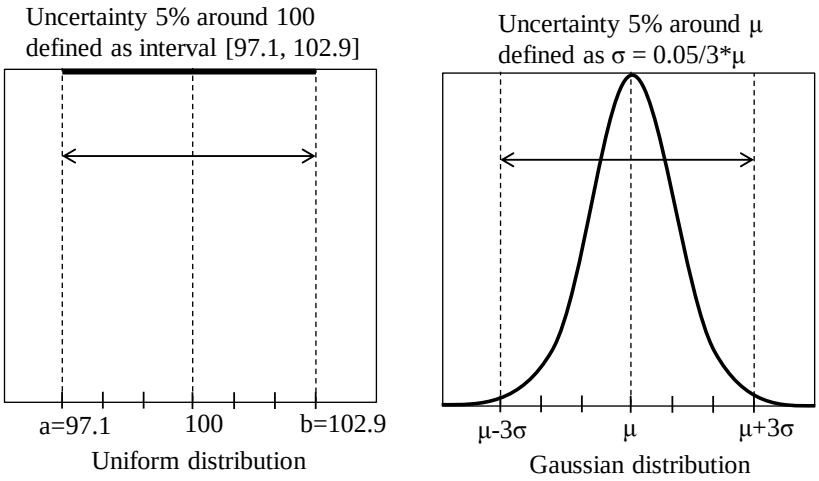


Figure 4.2: (a) To inject 5% uncertainty for a parameter characterized by a uniform distribution whose mean is 100 for example, we generate samples from a uniform distribution defined on the interval $[a = \mu - 0.05 \cdot \mu/\sqrt{3}, b = \mu + 0.05 \cdot \mu/\sqrt{3}]$. (b) The interval used for the case of a Gaussian distribution whose mean is μ.

(described later). This, in turn, may increase the computational runtime.

Uncertainty can be injected into the application or/and the architecture, depending on what design parameters are assumed to be affected by uncertainties and to what degree. This injection will be done in different amounts or degrees during the design space exploration depicted in Fig. 4.1. The injection process amounts to generating samples from pre-specified probability distributions during the Monte Carlo simulation technique used to evaluate reliability, execution time, and energy. Because we allow working with any type of probability distribution, we must define what is meant by *injecting a given percentage of uncertainty* into the design parameters of interest. We do that by pre-specifying the mean and the variance of the probability distributions out of which the sampling is done according to the rules listed in Table 4.1.

The rationale behind the rules presented in Table 4.1 can be explained

with the help of Fig. 4.2. For example, let us assume that the uncertainty is modeled for some design parameter with a uniform distribution. Then, modeling 5% of uncertainty in this design parameter during the design space exploration is achieved by having the MC simulation (discussed later in a different section) generate samples from an interval as shown in Fig. 4.2.a for the case when, for example, the mean is $\mu = 100$. That is because the variance (whose square root is the standard deviation, σ) is given by the expression $Var = (b - a)^2/12$. In the case of a Gaussian distribution, samples are generated randomly from a distribution $Gaussian(\mu, \sigma)$ but only samples falling inside the interval $[\mu-3\sigma, \mu+3\sigma]$, as shown in Fig. 4.2.b are accepted, which represent 99.7% of all generated samples. The case of the beta distribution is similar to that of the Gaussian case. The difference is only in the actual confidence level, which can be different from 99.7%. Note that similar rules can be derived for any other type of distribution that we may be interested in using to model parameter uncertainty. For simplicity, in this thesis, we restrict ourselves to using uniform and Gaussian distributions for modeling the execution time and the power consumption of architecture components and for modeling the transition probabilities inside the reliability model (discussed later). In addition, beta distribution is used to model failure rates of components, similarly to the study in [23]. However, our framework is flexible and can easily

Table 4.1: Rules for defining mean and variance of distributions from which sampling must be done to achieve a certain degree of uncertainty injection.

Probability Distribution	**Uncertainty** 1%	**Uncertainty** 5%	**Uncertainty** 10%
$Uniform(\mu, \sigma)$	$\sigma = 0.01 \cdot \mu_1/\sqrt{3}$	$\sigma = 0.05 \cdot \mu_2/\sqrt{3}$	$\sigma = 0.1 \cdot \mu_3/\sqrt{3}$
$Gaussian(\mu, \sigma)$	$\sigma = 0.01 \cdot \mu_1/3$	$\sigma = 0.05 \cdot \mu_2/3$	$\sigma = 0.1 \cdot \mu_3/3$
$Beta(\mu, \sigma)$	$\sigma = 0.01 \cdot \mu_1/3$	$\sigma = 0.05 \cdot \mu_2/3$	$\sigma = 0.1 \cdot \mu_3/3$

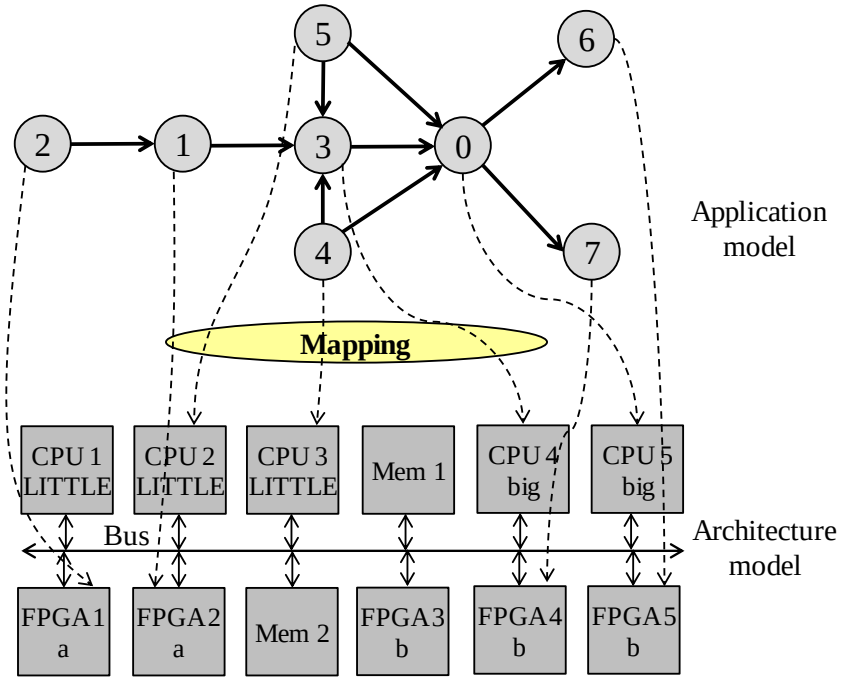


Figure 4.3: ABS application mapping problem. We only show the mapping of the tasks for simplicity.

accommodate other probability distributions if embedded designers find their data to fit better such distributions.

4.3 Application Modeling

To be able to formulate the mapping problem in a computer program like the one developed in this thesis, one must work with models for both *Application* and *Architecture* in Fig. 4.1. In this thesis, we adopt the notation from [50; 51; 52] and model applications using Kahn Process Networks (KPNs), which are among the most popular models of computation used in embedded systems design [27; 33]. A KPN is represented as an application directed graph $G_{AP}(V_{AP}, E_{AP})$. Each node or vertex $v_i, i \in \{1, .., |V_{AP}|\}$ corresponds to a process or task of G_{AP}. For each vertex v_i, we define $B_i = \{e_j \in E_{AP}\}$ to be the set of application channels

connected to vertex v_i. When a vertex is mapped to a hardware component, ht_i represents the hardware execution time. When the task can be executed on multiple hardware cores, ht_i becomes a set $ht_i = \{ht_{i1}, ht_{i2}, .., ht_{iU}\}$, where U is the number of hardware cores on which the task can be executed. When a vertex is mapped to a software component, st_i is the software execution time. When the task can be executed on multiple software components, st_i becomes a set $st_i = \{st_{i1}, st_{i2}, .., st_{iV}\}$, where V is the number of software components on which the task can be executed. Each edge $e_j, j \in \{1, .., |E_{AP}|\}$ corresponds to a data or control link between two different tasks of G_{AP}. If a communication link is mapped onto a memory core, mt_j represents the memory access time, which will be added to the path delay. When the link can be mapped to multiple memory components, mt_j becomes a set $mt_j = \{mt_{j1}, mt_{j2}, .., mt_{jW}\}$, where W is the number of memory components on which the link can be mapped to.

For example, Fig. 4.3 shows the application graph of the ABS testcase. It includes 8 tasks and 9 communication channels. The graph G_{ABS} includes the nodes set $V_{ABS} = \{v_i | i \in \{1, 2, ..., 8\}\}$ and the edges set $E_{ABS} = \{e_j | j \in \{1, 2, ..., 9\}\}$. Since there are 5 hardware components in the architecture platform shown in Fig. 4.3, when a node v_i is mapped to a hardware component, the hardware execution time ht_i becomes a set $ht_i = \{ht_{i1}, ht_{i2}, ..., ht_{i5} | i \in \{1, 2, ..., 8\}\}$. Similarly, since there are also 5 software components in the target architecture for the ABS application, when a vertex is mapped to a software component, the software execution time st_i becomes a set $st_i = \{st_{i1}, st_{i2}, ..., st_{i5} | i \in \{1, 2, ..., 8\}\}$. Likewise, if a communication link is mapped onto a memory core, the memory access time mt_j becomes a set $mt_j = \{mt_{j1}, mt_{j2} | j \in \{1, 2, ..., 9\}\}$.

4.4 Architecture Modeling

The architecture model is also represented by a graph $G_{AR}(V_{AR}, E_{AR})$, where the sets V_{AR} and E_{AR} denote the architecture components and the connections between them. The set of architecture components consists of two disjoint subsets: the set of processing cores (P) that include hardware and software elements and the set of memories (M), $V_{AR} = P \cup M$. The delay of a communication link between two different architecture components is denoted as lt_{pq}, with $p, q \in \{1, .., |E_{AR}|\}$. The power dissipations are denoted as p_c for the core c during execution, as p_m for the memory core m, and as p_l for the communication links. In this thesis, we assume that the architecture platform is given because we do not address the problem of architecture synthesis.

For instance, Fig. 4.3 also shows the architecture platform, onto which the ABS application will be mapped. It includes 5 hardware components (e.g., FPGAs) and 5 software components (e.g., CPUs). These architecture components communicate through the common bus link. The software (SW) components are represented by general central processing units (CPUs) but can also include (application specific) digital signal processors (DSPs) as well. These are referred to as "software" because they are supposed to run application tasks compiled into software executables that will be run as programs. Components like field programmable gate arrays (FPGAs) and application specific integrated circuit (ASICs) are referred to as hardware (HW) components. Memories represent the third category of components, to which application communications can be mapped to.

4.5 Generation of Initial Candidates

With regard to Fig. 4.1, the GA algorithm requires a set of limited solutions. This is indicated with the corresponding "Generate initial candidates" block in Fig. 4.1. Therefore, in the proposed method, an initial set of candidate solutions needed by the evolutionary algorithm (discussed later in this chapter) is first generated. We randomly generate the initial set of candidate solutions for simplicity. The idea is to generate a starting point that captures specific requirements on the amount of hardware resources used and which is not much different from a design solution arrived at via a completely manual approach. Because the hardware platform is fixed, the initial candidate solutions represent different mappings of the application on to the hardware platform.

4.6 Design Space Exploration Using Genetic Algorithms

The "Design Space Exploration" block from Fig. 4.1 is where new solutions are generated and where design optimization takes place. This is a challenging step not only because of the complexity of the mapping problem but also because it must accommodate uncertainty as well. We propose to use the stochastic optimization algorithm: Non-dominated Sorting Genetic Algorithm (NSGA-II) to solve this problem. The NSGA-II is designed and instrumented to guide the search process toward robust and optimal solutions and is achieved by closing the outer loop shown in Fig. 4.1, hence implicitly taking into consideration the searched solution's reaction to uncertain parameter variations.

The mapping problem is a multi-objective optimization problem whose objective functions or *quality attributes* often conflict. In this thesis, we consider the three objectives described next.

4.6.1 Objective 1: Reliability

The first objective function is the reliability of the system, which needs to be maximized. To estimate reliability, we adopt the approach employed in [53] and describe it next. This reliability model is based on absorbing discrete time Markov chain (DTMC) models, which have been used for a long time [54]. A DTMC model is a graphical model consisting of a finite state machine like state graph. For a given mapping solution, the DTMC model is constructed from the architecture platform of the system. A node in this graph represents the execution of an application task mapped to that component. An arc represents the transfer of execution between tasks mapped to different components. The graph has added a super-initial node to represent the execution start of the application. In addition, arcs are added to the graph from the newly added node; these arcs are labeled with appropriate initialization probabilities $q_0(c_i)$. A super-final node is also added to capture the end of the execution of the application. The DTMC model is characterized by a transition probability matrix $P = [p_{ij}]$. The probability p_{ij} represents the probability that task j is called after executing task i. This model assumes that the components of the system fail independently. Moreover, the probability R_i that the component performs its function correctly characterizes the reliability of the component c_i. In other words, this is the probability that that component finishes its task correctly and it also transfers the control to the

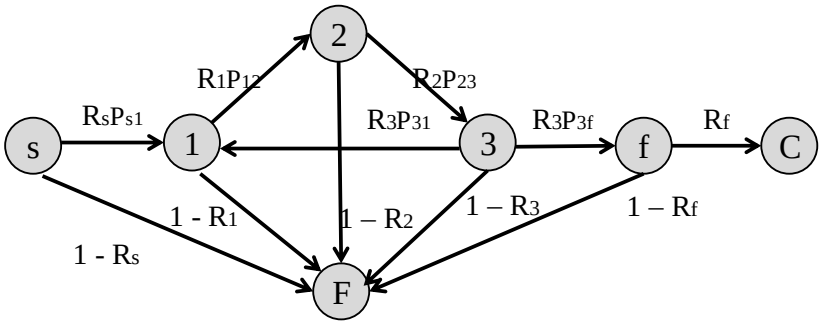


Figure 4.4: DTMC model when states C and F are added.

next task without a failure.

The DTMC graph model is modified further by adding two absorbing states C and F. The meaning of these new states in the graph model is that they represent the correct output and failure cases during the execution of the application. These new states require the transition probability matrix P to be modified into $\hat{P}$. In this modification, the original transition probability p_{ij} is replaced with $R_i p_{ij}$. The new notation $R_i p_{ij}$ denotes the probability of correct output from task i and of successful transfer of control to task j. To denote the correct execution, a new arc is created from the final state n to the state C. This arc is annotated with transition probability R_n. Similarly, a directed arc from state i to state F is added to represent the failure of a component c_i. This arc is annotated with transition probability $(1 - R_i)$. Once the DTMC is constructed as described above, the reliability of the application mapped to the architecture platform can be estimated as the probability of reaching the absorbing state C. An example of the DTMC model is shown in Fig. 4.4.

Now, let us denote with Q the matrix derived from $\hat{P}$ after the rows and columns corresponding to the absorbing states C and F are deleted. In this case,

the probability of reaching state n from 1 through k transitions is represented by $Q_k(1, n)$. Note also that the number k of such transitions from the initial state 1 to the final state n can vary up to infinity. Under these assumptions, it can be shown [54; 53] that the infinite summation converges as given by the following equation (I is the identity matrix),

$$S = I + Q + Q^2 + Q^3 + ... = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1} \tag{4.1}$$

In the above equation, S is called the fundamental matrix of the DTMC. $S(i, j)$ is the expected number of visits to state j starting from state i before it is absorbed. It was used by the authors of [54] to derive an expression to estimate the *architecture based reliability* of the overall system as follows:

$$R = S(1, n)R_n \tag{4.2}$$

In this thesis, we use this reliability estimation method. Normally, reliability is desired to be maximized. However, in order to construct the three-dimensional solution space such that each metric improves as we move towards the center of the coordinate system, we need to transform the maximization problem into a minimization problem. The objective of maximizing the reliability of the system can be written as a minimization objective as follows:

$$\min \{1 - R\} \tag{4.3}$$

Note that other reliability models can be used here as well. Our framework is generic enough and not restricted to using only DTMC based reliability models; it can employ any reliability model of interest such as that in [24] for example.

However, in this thesis we use the DTMC model for simplicity.

4.6.2 Objective 2: Execution Time as Measure of Performance

The second objective function is the one that minimizes the maximum execution or processing time of the critical path from the set of all paths (set denoted as $Path$) inside the application task graph. This minimum value is used as a direct measure of performance, and using the notations introduced in the previous sections, can be expressed as follows.

$$\min \left\{ \max_{Path} \{ \sum_{i \in V_{AP}, i \in Path} ht_{iu} x_{iu} + \sum_{i \in V_{AP}, i \in Path} st_{iv} x_{iv} + \sum_{j \in E_{AP}, j \in Path} [\ lt_{kl} + (mt_{jw} + lt_{mn}) x_{jw}\ ] x_j\ \} \right\} \tag{4.4}$$

The first term in the above equation represents the contribution of the hardware cores to the execution time of the critical path. Similarly, the second term captures the contribution from the tasks executed as software modules. Finally, the third term is the contribution to the processing time of the delay due to direct links between different architecture cores and possibly of the memory access time if the application communication channel j is mapped onto a memory core. Here, mt_{jw} is the memory access time with $w \in \{1, .., |M|\}$, lt_{kl} is the link delay between architecture cores k and l with $k, l \in \{1, .., |V_{AR}|\}$ and lt_{mn} is the link delay between architecture cores m and n also with $m, n \in \{1, .., |V_{AR}|\}$.

The variables x_{iu}, x_{iv}, x_{jw}, and x_j are *decision variables* that capture whether a task i is mapped to a hardware core u or a software core v, whether

a communication channel j is mapped to a memory core w, and whether a communication channel is contained within a core (i.e., two communicating tasks are mapped to the same core, in which case $x_j = 0$) or not. The values of these decision variables are different for different mapping solutions, which are generated during the genetic algorithm based design space exploration from Fig. 4.1.

4.6.3 Objective 3: Energy Consumption

The third objective function minimizes the energy consumption of the whole system. It is given by:

$$\min \Bigg\{ \sum_{i \in V_{AP}} ht_{iu} p_u x_{iu} + \sum_{i \in V_{AP}} st_{iv} p_v x_{iv} + \sum_{j \in E_{AP}} [\ lt_{kl} p_k + (mt_{jw} p_w + lt_{mn} p_n) x_{jw}\ ] x_j \Bigg\} \tag{4.5}$$

The energy consumption of the whole system consists of the energy consumption of the processing cores, the communication links, and the memories. The energy consumption of the processing cores can be represented by the sum of the energy consumption for the hardware cores and the software cores. Similarly, the energy consumption of the communication links can be captured by the product of the execution time spent on communication and the power dissipation for the communication links. Likewise, the energy consumption of the memories is the product of the total processing time of the memories and the power dissipation for memory cores. In the above equation, The first and the second terms represent the contribution of the processing cores to the energy consumption. Similarly,

the third term captures the contribution from the communication links and the memories to the energy consumption.

4.6.4 Solving the Multi-objective Problem

Once all three objective functions are defined as discussed in the previous sections, the overall optimization problem − which in our case is the mapping problem − can be written in a generalized form as follows [55]:

$$\min_{\mathbf{x}} \quad \mathbf{z} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))^T \tag{4.6}$$

$$\text{s.t.} \quad \mathbf{x} \in X \tag{4.7}$$

In the above equation, $\mathbf{x}$ represents a particular solution, and X is a set of feasible solutions. In our case, a mapping solution is captured by the individual decision variables discussed earlier that completely describe how application tasks are assigned to the cores of the architecture platform. The three individual objective functions f_1, f_2, and f_3 effectively evaluate the expressions from equations (4.3), (4.4), and (4.5) for a given mapping solution. The overall objective function $\mathbf{z} = f(\mathbf{x})$ translates a solution $\mathbf{x}$ from the *decision space* defined by the decision variables to a point in the *objective space* defined by the three objective or cost functions. In our case, the objective space is three dimensional, as shown in Fig. 1.2, and the overall objective function is defined as the equally weighted summation of the three individual objective functions.

Because multi-objective optimization problems usually do not have a single best solution which optimizes all objectives at the same time, we are interested in finding a set of solutions that form the so called *Pareto frontier*. The solution points that form the Pareto frontier are points that are *non-dominated* by any other solution point among all solutions from the feasible set. There has been a lot of work done on the topic of multi-objective optimization problems. One of the most popular approaches to solving this problem and to find the Pareto frontier is to use evolutionary algorithms due to their inherent ability to handle multiple objectives at the same time. Therefore, in this thesis, we construct our solution to the multi-objective mapping problem also using such an approach. More specifically, we use the Non-dominated Sorting Genetic Algorithm (NSGA-II) [56] because it was shown to offer benefits over other types of evolutionary algorithms [55] including ease of implementation and lower computational complexity. This algorithm implements the outer loop from Fig. 4.1. The pseudocode description of this algorithm is shown in Algorithm 1. For complete details about NSGA-II, please see [56].

The idea of the genetic algorithm is to iteratively generate new children solution populations from previous parent solution populations. This generation is usually realized using different forms of crossover and mutation. Details about NSGA-II based DSE will be explained in the following subsections.

Algorithm 1 Design space exploration based on NSGA-II

Inputs: N size of the population, M maximum number of generations.
Outputs: Pareto frontier, as non-dominated solutions in P_M.
P_0 = GenerateInitialPopulation(); // size N
$Q_0 = \emptyset$; // start with children set empty
EvaluateObjectiveFunction(P_0); // calculate fitness
RankPopulation(P_0); // done according to fitness values
for $(i = 0$ to $M - 1)$ **do**
    Q_i = SelectionCrossoverMutation(P_i); // create children population
    EvaluateObjectiveFunction(Q_i); // uncertainty aware, Monte Carlo based
    P_{i+1} = CombineParentsAndChildren(P_i, Q_i);
    RankPopulation(P_{i+1});
    P_{i+1} = SelectNIndividuals(P_{i+1}); // elitism: keep non-dominated
end

Encoding and Initial Population

In this thesis, we encode the mapping solution from the target application to the architecture platform as a string of integers. Each genotype (or representation of the possible mapping) consists of task nodes that can be mapped to SW components, task nodes that can be mapped to HW components, and communication arcs that can be mapped to memory components. Each gene in the chromosome (or genotype) represents a unique identifier of the component in the architecture platform. In addition, it has its own feasible set. For example, for genes representing nodes that must be mapped to SW components, only the set of CPUs in the architecture model form the feasible set.

In our NSGA-II, the chromosomes in the initial population are generated randomly. Moreover, we limit the size of the initial population and also the size of the set of generated individuals during each evolution process of the NSGA-II to reduce the runtime.

Fitness Function

The fitness function is defined for measuring the quality of mapping solutions. During the iterative process of the outer loop in Fig. 4.1, the fitness of the new solutions are evaluated by $EvaluateObjectiveFunction()$, which essentially uses equation (4.6). It is also this evaluation step that distinguishes our approach from previous work. Here, we assume uncertainties to affect design parameters. We capture such uncertainties as described in the previous sections. The evaluation step employs a Monte Carlo simulation technique to deal with uncertain quantities and will be discussed in the next section.

Selection

During each evolution generation of the NSGA-II, a proportion of the existing population is selected to be in the parents population. We use a tournament selection operator based on the rank and crowded distance of each solution in the population. The calculations of the rank and crowded distance are similar to [56].

Genetic Operators

We use the simulated binary crossover (SBX) operator and polynomial mutation in NSGA-II [56]. The crossover probability is $p_c = 0.9$ and the mutation probability is $p_m = 1/n$, where n is the number of decision variables.

Termination

In terms of the stopping conditions for our NSGA-II, a maximum number of generations is adopted to guarantee that the evolution process will stop.

Computational Complexity

The computational complexity of the NSGA-II algorithm is $O(MN^2)$, where M is the number of objectives and N is the population size. Because we are using three objectives, the runtime is longer than when one would focus on only one or two objectives. However, to keep the runtime under control one can adjust control several other factors like the population size that could be reduced a little and have a significant impact on the runtime because of the squared relationship. We report results of the runtime investigation of our tool in the Simulation Experiments in Chapter 5.

4.7 Selection

During the iterations of the genetic algorithm, design solution points are evaluated with the MC based technique. Some of those solution points should be selected and provided as output of the method. We want these solutions to be the most robust solutions. This is illustrated in Fig. 4.1 by the Selection box, where solutions that are found to be better than previous solutions are added to the list of robust solutions. This iterative process continues until the stopping criterion - based on computational runtime limits or solution quality - is met.

4.8 Estimation Under Uncertainty

To estimate different quality attributes of a solution candidate when input parameters are subject to uncertainty, we use a Monte Carlo (MC) simulation technique because the MC method is the tractable method that is capable of accommodating multiple types of probability distributions during its sampling process and because analytic solutions are extremely difficult or impossible to derive when dealing with a wide variety of probability distributions. During MC runs, each candidate solution is evaluated for different values or samples from the input distributions that characterize uncertain parameters and this evaluation process is agnostic to the assumptions made about the input distributions.

During the NSGA-II genetic algorithm based DSE depicted in Fig. 4.1, each new solution candidate must be evaluated in order to estimate three different design attributes of interest. These attributes are reliability, execution time, and energy. Their deterministic calculation, in a traditional design flow, can be done using the main expressions from equations (4.3), (4.4), and (4.5). If however, we assume that design parameters are affected by uncertainties, then, any of the design attributes that is affected by uncertainties cannot be estimated anymore using deterministic equations. Analytic solutions are extremely difficult or impossible to derive when dealing with a wide variety of distributions. Instead, estimation techniques that model and can handle uncertainty must be employed. Usually, in such situations, a Monte Carlo simulation based technique represents the only tractable method that is capable of accommodating multiple types of probability distributions [23; 38; 41]. Therefore, in this thesis, we use Monte Carlo

simulation based techniques to estimate the attributes of interest that are affected by uncertainty. Such a formulation allows us to combine quality attributes which inherit uncertainty with ones that are deterministic (or certain) in any mixture.

4.8.1 Consideration of Uncertainty Correlations

When modeling the uncertainty in design parameters in embedded systems, we need to consider the uncertainty correlations. The reason for that is because multiple uncertainty sources (e.g., some heat sources) may affect several components simultaneously, thus, a correlation between uncertainties of components exists. The work in [24] is a recent attempt to consider uncertainty correlations when modeling components' uncertainties. We model the uncertainty correlations in a similar way, but with the difference that we consider the uncertainty correlations under the influence of multiple uncertainty sources. We do that because in practice, the architecture components are usually affected by multiple uncertainty sources, and the uncertainty in design parameters are the combination of the influence of different uncertainty sources. First, we need to introduce some of the definitions that will be used later.

Correlation Group: A simple set of components being affected by the uncertainty sources. Such as Fig. 4.5 group A.

Independent Group: If the correlation group g contains only one component, and group g is affected by the same uncertainty source, then, we call group g an independent group. Such as Fig. 4.5 group B.

Single Correlation Group: If all the components in the correlation group

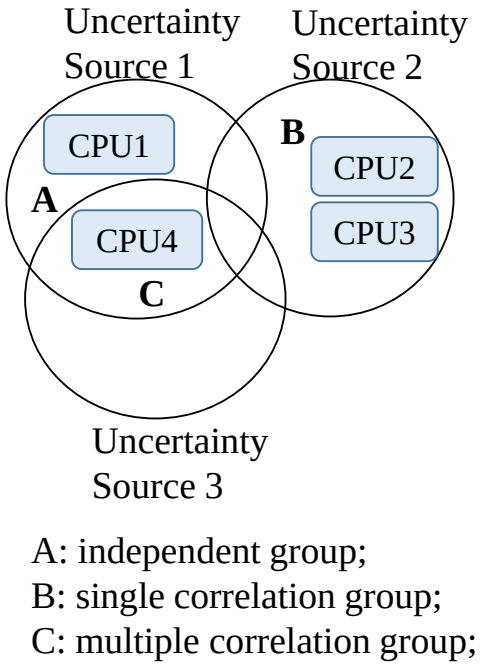


Figure 4.5: Definitions of different correlation groups.

g are affected by the same uncertainty source, then, we call group g a single correlation group. Such as Fig. 4.5 group C.

Multiple Correlation Group: If **some** of the components in the correlation group g are affected by multiple uncertainty sources, then, we call group g a multiple correlation group.

q_g^{th} *percentile:* is the percentile we sample in correlation group g from its uncertain parameters probability distributions. It is used to guarantee that all components in group g vary together for each sampling process.

The consideration of uncertainty correlations can be achieved in the following way:

(1) For an independent group, samples from the uncertain design parameters probability distributions are generated using independently sampled parameters.

(2) For a single correlation group, samples from the uncertain design parameters are generated using the q_g^{th} percentile of its probability distribution, where q_g is a uniformly distributed random number that satisfies $q_g \in [0, 1]$ for each sampling process.

(3) For a multiple correlation group, samples from the uncertain design parameters probability distributions are generated using the q_g^{th} percentile of its probability distribution, where q_g^{th} is determined using the following equation:

$$q_g^{th} = \sum_{i=1}^{n} \alpha_i \cdot q_g^i \tag{4.8}$$

where α_i is the coefficient (it can be set by the user for simplicity) between [0,1], which is used to measure the degree of influence from the uncertainty source i to group g, and it satisfies $\sum_{i=1}^{n} \alpha_i = 1$. q_g^i is the q_g^{th} percentile for group g affected by uncertainty source i.

Therefore, by generating samples in this way, the uncertain parameters from the components in a single correlation group vary together, and their variations are independent from those of different single correlation groups. Also, the samples of the uncertain parameters for those components in multiple correlation groups combine the influence of all uncertainty sources, and such combinations are independent of those of other multiple correlation groups.

4.8.2 Consideration of Different Levels of Uncertainty

With the concept of correlation groups and the proposed uncertainty modeling defined, we can easily consider different levels of uncertainty. We inject different levels of uncertainty into the design parameters of the components according to

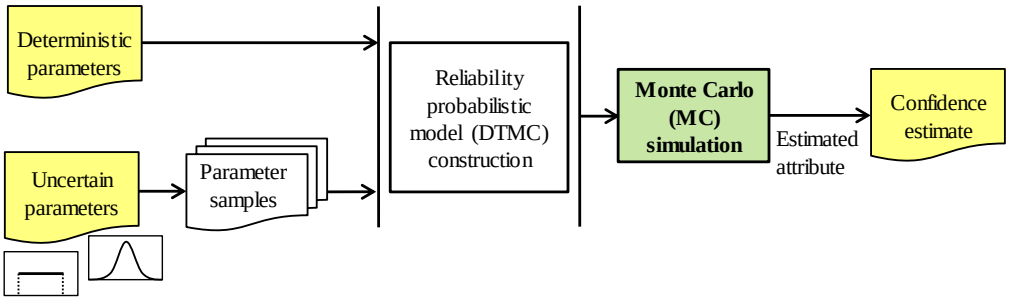


Figure 4.6: Block diagram of the Monte Carlo simulation based technique to estimate reliability.

the assumed degree of the influence from the uncertainty sources forming the correlation groups. For instance, if one correlation group is close to some heat source, that may translate in an actual 10% level of uncertainty in design parameters for the components in that correlation group. That means we must inject 10% level of uncertainty for the design parameters of the components in that correlation group. Similarly, components in a different correlation group may be far from such heat sources, and consequently, they may need to only have 1% level of uncertainty injected. Details as to how to inject different levels of uncertainty into the design parameters have been discussed in the uncertainty modeling section.

4.8.3 Robust Estimation of the Reliability

To estimate reliability, we employ the enhanced Monte Carlo estimation technique proposed in [23; 38] together with the consideration of both uncertainty correlations and different levels of uncertainty. This technique is represented by the *Monte Carlo Simulation* block in Fig. 4.1 and detailed in Fig. 4.6.

Input Parameters

The input to the *Monte Carlo Simulation* block from Fig. 4.6 is the probabilistic DTMC reliability model, which includes deterministic parameters and uncertain parameters. The deterministic parameters can be obtained from the traditional embedded systems design parameters, while the uncertain parameters can be achieved through the uncertainty injection process. As a means to capture heterogeneous uncertainties in design parameters, uncertain parameters are characterized by generalized probabilistic distributions. Details about the uncertain parameters were described in Section 4.2.

Reliability Probabilistic Model (DTMC) Construction

The next step of the robust estimation of the reliability is to construct the probabilistic model. This probabilistic model is based on absorbing discrete time Markov chain models, which have been described in Section 4.6.1. Given the fact that the inputs are probability distributions, the resulting evaluation model parameters become probability distributions or functions of probability distributions.

Monte Carlo (MC) Simulation

During the Monte Carlo iterations, these distributions are sampled with consideration of both uncertainty correlations and different levels of uncertainty, to generate instances that are then used as numerical values to compute the attribute of interest. Once the samples are obtained from the input distributions, we update all the corresponding evaluation model parameters and recompute the reliability

estimate. In this way, the impact of uncertainties on the estimation process is captured.

One single run of MC simulation leads to one numerical value of the reliability estimate. Due to the uncertain parameters, the estimated reliability from different MC runs are most often not identical. The estimated reliability metric becomes a variable quantity itself whose distribution is unknown. The variation of this quantity will represent an important measure that summarizes the impact of uncertainties.

Robust Estimates

To indicate the robustness of the reliability estimate, we use percentiles obtained from the MC runs. Since this has to be done without knowing the distribution, we use a non-parametric statistical estimation technique similar to [23]. This approach builds on previous results that use probabilistic quality models for the attribute of interest a − in this case reliability − and statistical estimation techniques to derive a single measure of interest, $\hat{a}$ (such as expectancy, variance, worst case value, confidence bound, etc.), as a descriptive measure of the extent and characteristics of uncertainty in the values in $A = \{a_i, i = 1, 2, ..., N\}$, whose distribution is unknown.

The accuracy of the estimate $\hat{a}$ depends on the number of MC runs, i.e., the sample size. However, to keep the computational runtime reasonable, we employ a dynamic stopping criterion based on accuracy monitoring, which works by using a sliding window of a minimum of k MC runs $(a_1, a_2, ..., a_k)$. For each *snapshot* of

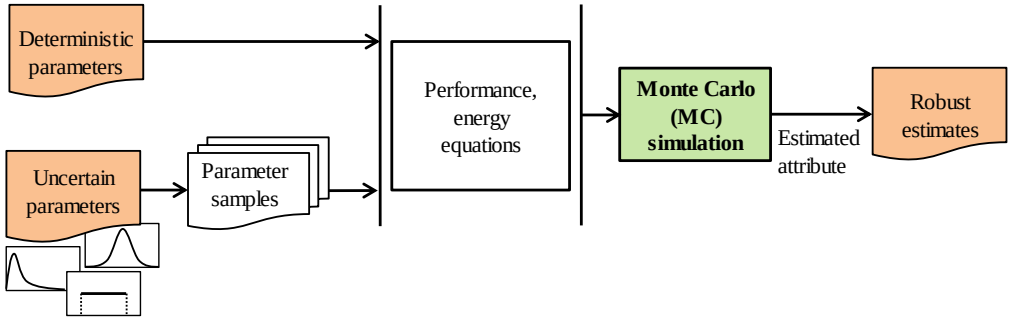


Figure 4.7: Block diagram of the Monte Carlo simulation based technique to estimate execution time, and energy consumption.

the sliding window, one of the above estimation methods is used to compute an $\hat{a}$. In this way, the sequence $\hat{A} = \{\hat{a_1}, \hat{a_2}, ..., \hat{a_k}\}$ is constructed and monitored. Only the last k samples of the estimate $\hat{A}$ are monitored. A statistical significance test is done on the samples of $\hat{A}$, and the relative error of the estimate $\hat{A}$ is checked against a tolerance level (i.e., 0.05). Once the error is smaller than the desired tolerance level, as the objective is to detect if sufficient accuracy has been obtained, the Monte Carlo runs are stopped.

4.8.4 Robust estimation for Performance and Energy Consumption

To estimate the performance and energy consumption attributes, the Monte Carlo simulation technique is simpler because here we do not need to build the probabilistic DTMC model. During multiple MC runs, parameters affected by uncertainties are also sampled from their respective probability distributions and used as numerical values inside equations (4.4) and (4.5) as Fig. 4.7 illustrates.

As mentioned earlier, in this thesis, we assume increased design uncertainties due to variations in fabrication processes, supply voltage, and temperatures,

which have been discussed and modeled in recent literature. This uncertainty increases as we go to deeper nanometer technology nodes. To model that, in the simulation results section, we will investigate different degrees of uncertainty. By increased uncertainty that we inject we mean that some design parameters or variables are even less accurately known.

4.9 Robustness of Design Solution Points

Generally, the output of the Monte Carlo simulation technique to estimate a certain attribute of interest for a given mapping solution is a number of samples out of the probability distribution that characterizes the unknown attribute. We use the 95 percentile estimate as the actual value used to generate and plot the robust Pareto frontier in the objective space. Working with percentile estimates provides a means to quantify or specify the robustness of the solution. The higher the percentile, the more robust the given solution is against uncertainties. Robustness is defined as the ability of a given solution to be immune or to tolerate uncertainties while still guaranteeing the desired performance.

Aside from generating the robust Pareto frontier in the three-dimensional objective space *(1-reliability) vs. performance vs. energy*, during each of the genetic algorithm iterations (see Fig. 4.1), solution points that are found to be better than previously found solutions are selected and added to the list of *best robust solutions*. This is a short list of potential design solution points from which embedded systems designers may select a final solution.

4.10 Conclusions

This chapter presented a design methodology for solving the mapping problem in embedded systems under uncertainties. The proposed method was incorporated within an automation software tool to integrate uncertainty models and novel optimization algorithms constructed with Monte Carlo and evolutionary algorithms. Details about the simulation experiments will be discussed in the next chapter.

CHAPTER 5

Simulation Experiments - Comparison to Traditional Method

Having described the theoretical concepts that contribute to the uncertainty aware mapping of embedded systems design in the previous chapters, this chapter presents the comparison experiments between the traditional point-estimate method and the proposed robust-estimate method for the Motion-JPEG (MJPEG) and the MP3 testcases. This chapter is organized into four sections. The first part of this chapter briefly introduces the experimental setup and the testcases. The second section describes the architecture platform we use during the simulation experiments. In the third section, we compare results obtained with two different estimation techniques: the proposed robust approach and the traditional deterministic approach. In the last section, we compare the Pareto frontiers generated by the proposed robust approach and the deterministic approach for both MJPEG and MP3 testcases.

5.1 Experimental Setup and Testcases

We report simulation results obtained with both the proposed design method and the traditional design method. For the NSGA-II genetic algorithm implementation, we integrate into our tool the C implementation publicly available at [57]. All simulations are done on a 64 bit Intel i5-4690 CPU, 3.50 GHz x4 with 8 GB memory running the Ubuntu 14.04 LTS operation system. Both the traditional

Table 5.1: Parameters of NSGA-II.

Parameter	**Value**
initial population size	256
number of generations	512
crossover probability	0.8
mutation probability	0.2

deterministic approach and the proposed robust approach have been integrated into the software framework that we developed, and which we call the Design of Embedded Systems Under Uncertainty (DESUU) tool. The parameters of the NSGA-II genetic algorithm we have used for DSE are the same as the one used by the Sesame simulator, and are listed in Table 5.1. Sesame [52], an abbreviation for "Simulation of Embedded Systems Architectures for Multi-level Exploration", is a system-level modeling and simulation environment which aims at efficient design space exploration of embedded systems. Sesame recognizes separate application and architecture models, where an application model describes the functional behavior of an application and the architecture model defines architecture resources and capture their performance and energy constraints. Sesame employs the traditional point-estimate approach when mapping an application model onto an architecture model.

We use two testcases as our benchmarks: a Motion-JPEG (MJPEG) encoder and an MP3 decoder. The MJPEG testcase contains 8 tasks and 18 communication channels, and the MP3 testcase includes 27 tasks and 52 communication channels. We adopt these two testcases from [28]. Figure 5.1 shows the KPN of the MJPEG and the MP3 testcases.

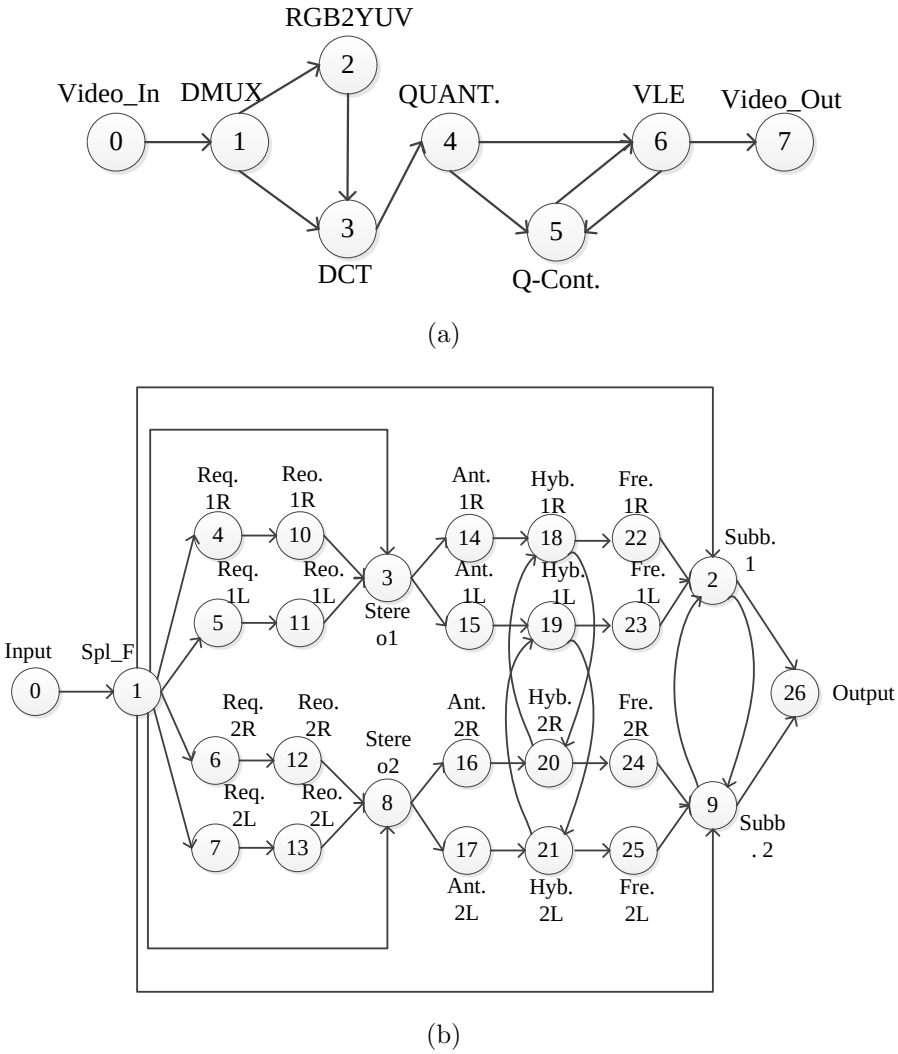


Figure 5.1: The KPN of (a) MJPEG testcase. (b) MP3 testcases.

5.2 Architecture Platform

We use the same target architecture as the Sesame simulator, which is a heterogeneous MPSoC including five different processors, connected to a shared bus and memory. The architecture platform is shown in Fig. 5.2. We have installed the Sesame simulator and collected the real simulated average execution cycles per task for both MJPEG and MP3 testcases. We use these values in the simulations with DESUU tool, to be able to do a fair comparison. Also, we adopt the power consumption values for different processors from [58].

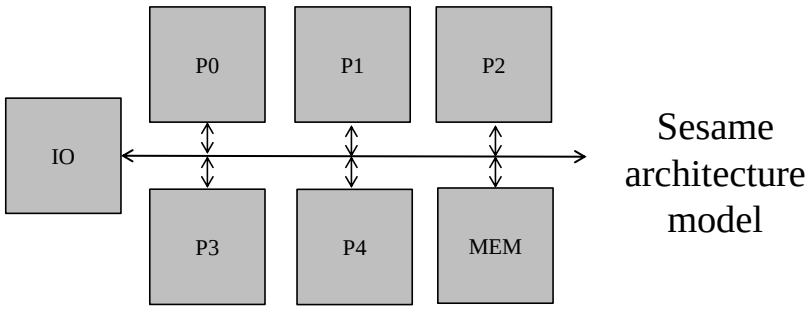


Figure 5.2: Architecture platform used in simulations done with the Sesame tool.

With respect to the target architecture, in order to mimic uncertainty resources, we assume that the I/O component generates more heat in time, which increases the temperature of the nearby components. The increase in temperature is modeled as an increase in the uncertainty in design parameters of the affected components. Thus, we inject 10% level of uncertainty for processors P0 and P3, 5% level of uncertainty for processors P1 and P4, and 1% level of uncertainty for processors P2 and the memory component.

To this end, we have installed and configured both the traditional deterministic approach (Sesame tool) and have developed the proposed robust approach (DESUU tool) to use the same NSGA-II parameters. Simulations are conducted on the same benchmarks and using the same architecture platform.

5.3 Robust and Deterministic Estimation

In the first set of simulations, we conduct a comparison of the estimation techiques used by the proposed approach (DESUU tool) and by the deterministic approach (Sesame tool). We look at the execution time and the energy consumption for MJPEG and MP3 testcases. For simplicity, we use a given round-robin mapping

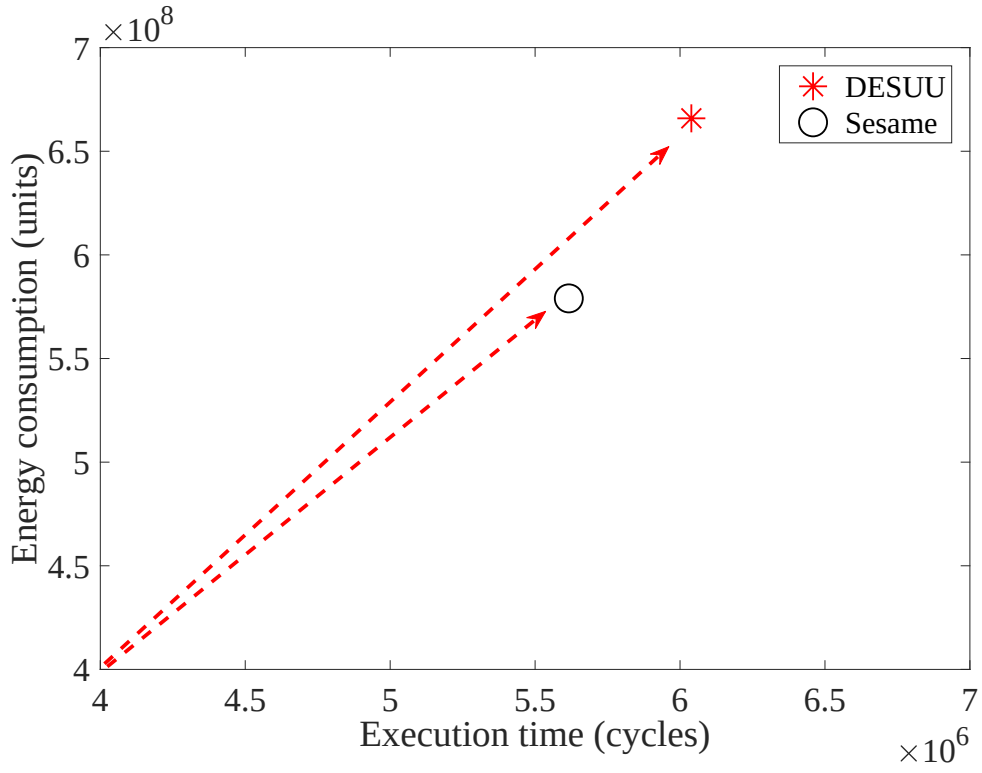
in both approaches. The execution time is characterized or measured by execution cycles and the energy consumption is described in nanojoule (nJ).

The comparison between the results obtained using the robust estimation approach and those obtained with the deterministic estimation approach is shown in Fig. 5.3. Recall that in the motivation example from Chapter 3, we saw that uncertainty in the design parameters of embedded systems may result or lead to different solutions from those found by traditional approaches. Here, Fig. 5.3. further shows that, for a given mapping, the Sesame tool provides smaller values for both performance and energy consumption. In contrast, the DESUU approach, which employs more robust and accurate technique provides larger values. Fig. 5.3 indicates that the Sesame approach underestimates the design attributes, which if used during the optimization process conducted during DSE, may lead to non-optimal final solutions.

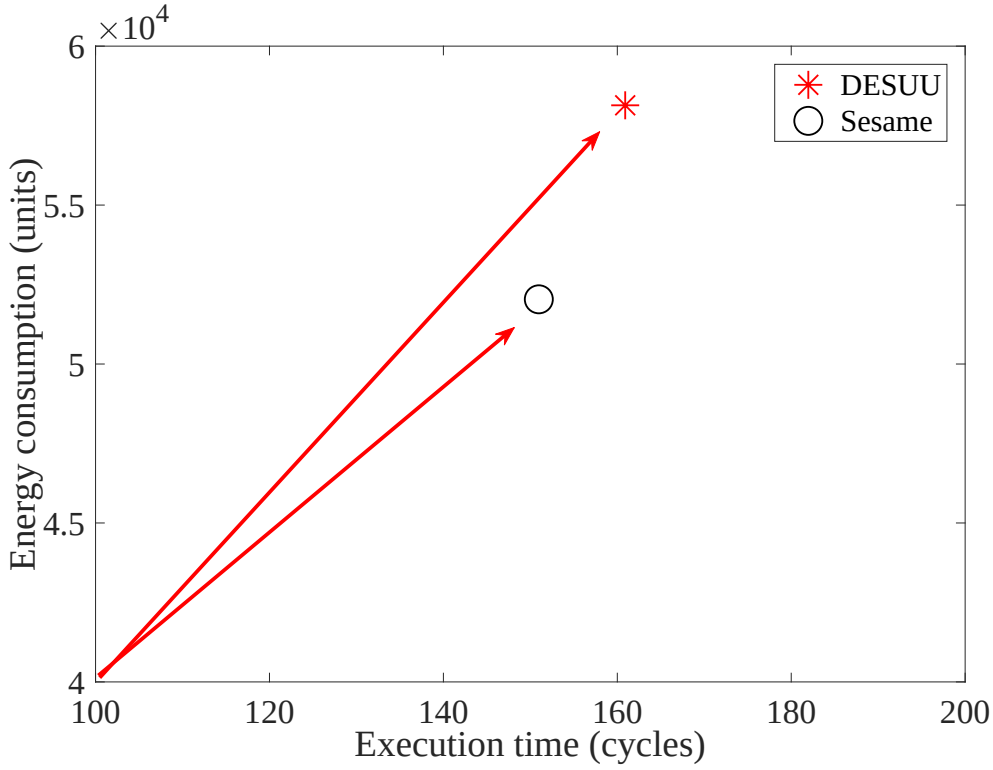
5.4 Pareto Frontiers

Next, we compute the Pareto frontiers obtained with both approaches. Fig. 5.4 shows that the proposed DESUU tool generates Pareto frontiers that are shifted away from those obtained by the Sesame tool. This shift is in aggrement with the results from the previous work. On those frontiers, the solution points that are closest to the system of coordinates are the solutions that represent the best trade-off between the execution time and energy consumption.

Furthermore, we observe that the solutions on the Pareto frontiers generated by Sesame approach appear to be better than the solutions on the Pareto

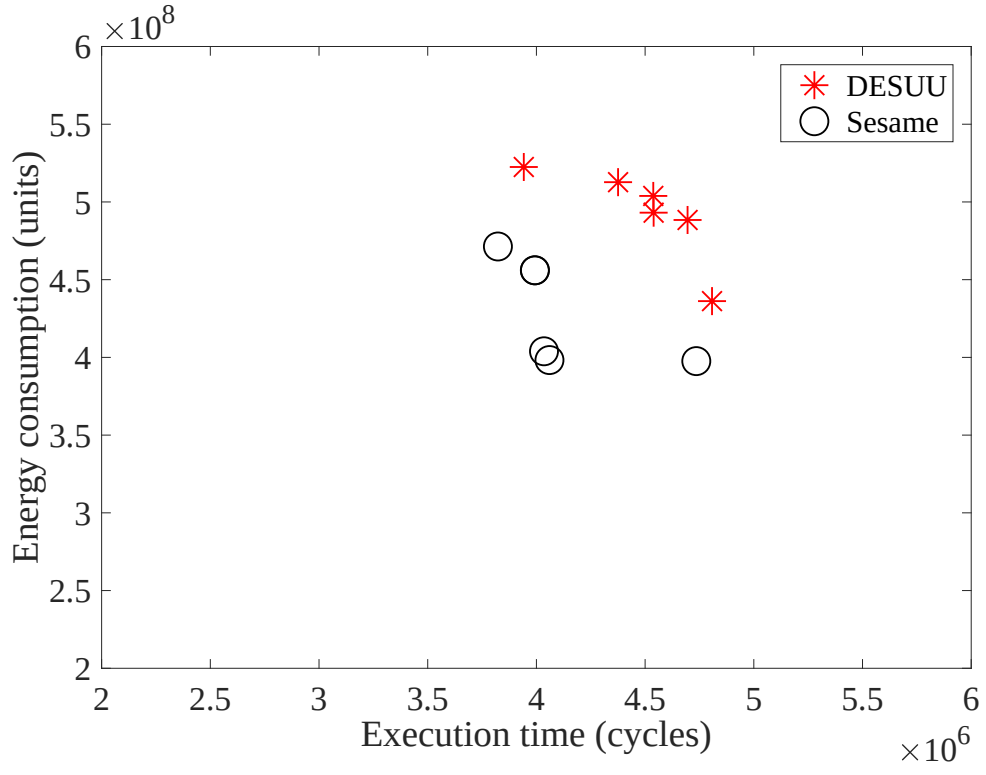


(a)

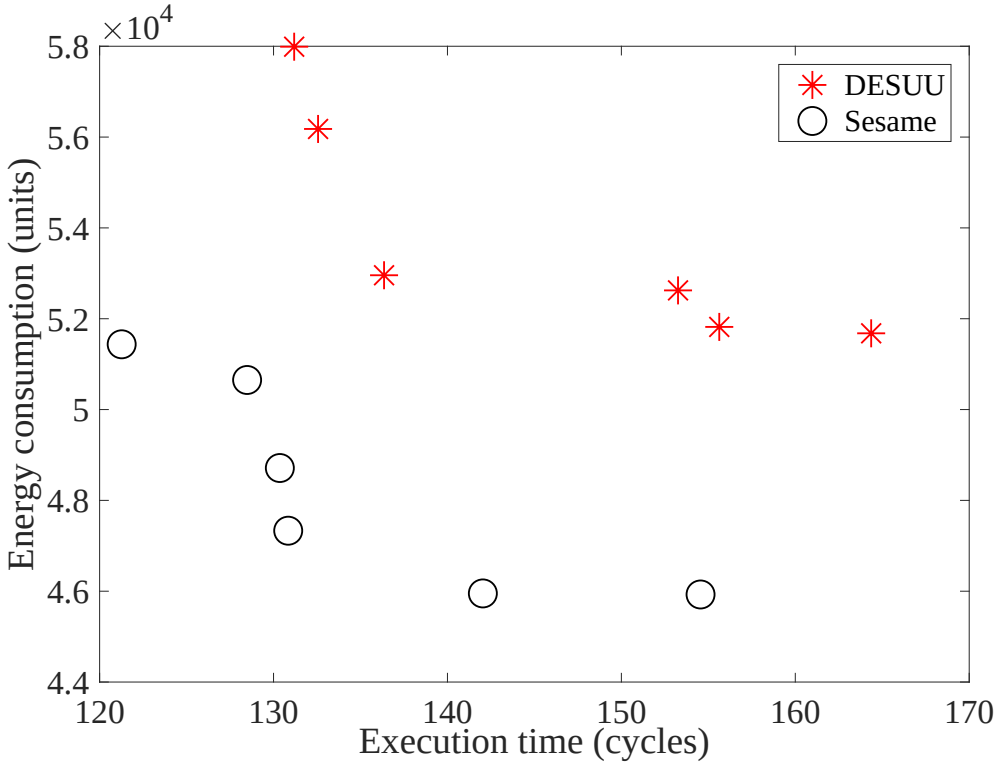


(b)

Figure 5.3: The comparison estimation between the Sesame approach and the DESUU approach for (a) MJPEG testcase (b) MP3 testcase.



(a)



(b)

Figure 5.4: Pareto frontiers generated by the Sesame approach and the DESUUU approach for (a) MJPEG testcase (b) MP3 testcase.

frontiers obtained by the DESUU approach. However, the simulations found by the Sesame tool are not aware of uncertainty and provide an optimistic view of what the performance and energy consumption values would be. We believe that the traditional deterministic approach of the Sesame tool under-estimates design attribute values, and that can lead to inaccurate final solutions.

5.5 Conclusion

In this chapter, we compared the Pareto frontiers generated by the proposed robust approach and the deterministic approach to the problem of mapping. These frontiers are in the two dimensional space characterized by the performance and the energy consumption attributes. Comparisons are done for two testcases: MJPEG and MP3. In the next chapter, we will report and discuss simulation results obtained with the proposed robust design method for a larger set of testcases. In addition, we will present a scalability analysis of the proposed DESUU tool.

CHAPTER 6

Simulation Experiments - Proposed Robust Method, Scalability Analysis

After conducting the comparison experiments between the traditional point-estimate method and the proposed robust-estimate method in Chapter 5, in this chapter, we report simulation results obtained with the proposed design method, which we implemented as a computer program in C++. As in Chapter 5, all simulations are done on a 64 bit Intel i5-4690 CPU, 3.50 GHz x4 with 8 GB memory running the Ubuntu 14.04 LTS operation system.

6.1 Testcases

We use four testcases as our benchmarks with their characteristics listed in Table 6.1. Here, we have split the tasks of each testcase into HW and SW modules randomly because we do not address in this thesis the problem of HW/SW partitioning. Therefore, for each testcase, HW tasks must be mapped to HW components and SW tasks must be mapped to SW components.

The first two testcases are from the automotive application domain, ABS (anti-lock break system) and ACC (adaptive cruise control). We adopted these two testcases from the study in [23]. The last two testcases are from the multimedia application domain, H.264 (video decoder) and JPEG (picture compression). We adopted the H.264 testcase from [59] and the testcase JPEG from [60]. The block

Table 6.1: Listing of the testcases used for simulations.

Testcase	**Num. of HW tasks**	**Num. of SW tasks**
ABS	5	5
ACC	6	5
H.264	5	5
JPEG	5	5

Table 6.2: Detailed description of the ABS testcase.

$Trans(c_i \rightarrow c_j)$	$P(c_i, c_j)$
$s \rightarrow 4$	$GAUSSIAN, 0.3, 0.01$
$s \rightarrow 5$	$GAUSSIAN, 0.3, 0.01$
$s \rightarrow 2$	$GAUSSIAN, 0.3, 0.01$
$s \rightarrow 0$	$GAUSSIAN, 0.1, 0.01$
$4 \rightarrow 0$	$GAUSSIAN, 0.7, 0.01$
$4 \rightarrow 3$	$GAUSSIAN, 0.3, 0.01$
$0 \rightarrow 6$	$GAUSSIAN, 0.5, 0.01$
$0 \rightarrow 7$	$GAUSSIAN, 0.5, 0.01$
$6 \rightarrow f$	1
$5 \rightarrow 0$	$GAUSSIAN, 0.7, 0.01$
$5 \rightarrow 3$	$GAUSSIAN, 0.3, 0.01$
$3 \rightarrow 0$	1
$2 \rightarrow 1$	1
$1 \rightarrow 3$	1
$7 \rightarrow f$	1

diagrams and their DTMC models used for reliability estimation of these testcases are shown in Fig. 6.1, 6.2, 6.3, and 6.4. The detailed descriptions of the DTMC models are listed in Tables 6.2, 6.3, 6.4, and 6.5.

We assume that a given testcase or application has been profiled and that the performance and power consumption numbers for all the tasks in the application graph when implemented on the SW and HW components of the application platform are known. These numbers are then treated as the mean values of the

Table 6.3: Detailed description of the ACC testcase.

$Trans(c_i \rightarrow c_j)$	$P(c_i, c_j)$
$s \rightarrow 1$	$GAUSSIAN, 0.2, 0.01$
$s \rightarrow 4$	$GAUSSIAN, 0.4, 0.01$
$s \rightarrow 7$	$GAUSSIAN, 0.4, 0.01$
$1 \rightarrow 2$	1
$2 \rightarrow 3$	$GAUSSIAN, 0.1, 0.01$
$2 \rightarrow 9$	$GAUSSIAN, 0.5, 0.01$
$2 \rightarrow 8$	$GAUSSIAN, 0.4, 0.01$
$3 \rightarrow f$	1
$4 \rightarrow 5$	$GAUSSIAN, 0.5, 0.01$
$4 \rightarrow 8$	$GAUSSIAN, 0.5, 0.01$
$5 \rightarrow 2$	$GAUSSIAN, 0.4, 0.01$
$5 \rightarrow 6$	$GAUSSIAN, 0.6, 0.01$
$6 \rightarrow 2$	1
$7 \rightarrow 5$	$GAUSSIAN, 0.5, 0.01$
$7 \rightarrow 8$	$GAUSSIAN, 0.5, 0.01$
$8 \rightarrow 9$	1
$9 \rightarrow f$	1

Table 6.4: Detailed description of the H.264 testcase.

$Trans(c_i \rightarrow c_j)$	$P(c_i, c_j)$
$0 \rightarrow 1$	1
$1 \rightarrow 2$	1
$2 \rightarrow 3$	1
$3 \rightarrow 4$	$GAUSSIAN, 0.5, 0.01$
$3 \rightarrow 5$	$GAUSSIAN, 0.5, 0.01$
$4 \rightarrow 6$	1
$5 \rightarrow 6$	1
$6 \rightarrow 4$	$GAUSSIAN, 0.5, 0.01$
$6 \rightarrow 7$	$GAUSSIAN, 0.5, 0.01$
$7 \rightarrow 8$	1
$8 \rightarrow 5$	$GAUSSIAN, 0.5, 0.01$
$8 \rightarrow 9$	$GAUSSIAN, 0.5, 0.01$

probability distributions that are used to model the uncertain values. The amount of uncertainty is controlled through the variance of the respective distribution as discussed in Chapter 4.

Table 6.5: Detailed description of the JPEG testcase.

$Trans(c_i \rightarrow c_j)$	$P(c_i, c_j)$
$0 \rightarrow 1$	1
$1 \rightarrow 2$	1
$2 \rightarrow 3$	1
$3 \rightarrow 4$	1
$4 \rightarrow 5$	$GAUSSIAN, 0.5, 0.01$
$4 \rightarrow 6$	$GAUSSIAN, 0.5, 0.01$
$5 \rightarrow 2$	$GAUSSIAN, 0.25, 0.01$
$5 \rightarrow 3$	$GAUSSIAN, 0.25, 0.01$
$5 \rightarrow 6$	$GAUSSIAN, 0.25, 0.01$
$5 \rightarrow 7$	$GAUSSIAN, 0.25, 0.01$
$6 \rightarrow 7$	1
$7 \rightarrow 8$	1
$8 \rightarrow 9$	1

6.1.1 Architecture Platform

Because reliability, performance, and energy consumption represent objective functions, the only constraints that we used in our problem formulation consist of the architecture platform being given and the HW/SW partitioning of the given application. Specifically, in our case we assume that the architecture platform has twelve components in order to be able to accommodate the largest application task graph that we investigated. That includes five software components, five hardware components, and two memory components. The communication arcs in the graph are assumed to be implemented via memory mapping; that is the source task writes into a memory component and the destination tasks read from the memory component. Our architecture is a hypothetical one, which we envisioned based on the projections made by the research community about multiprocessor systems-on-chip (MPSoCs); that future embedded systems will be composed of

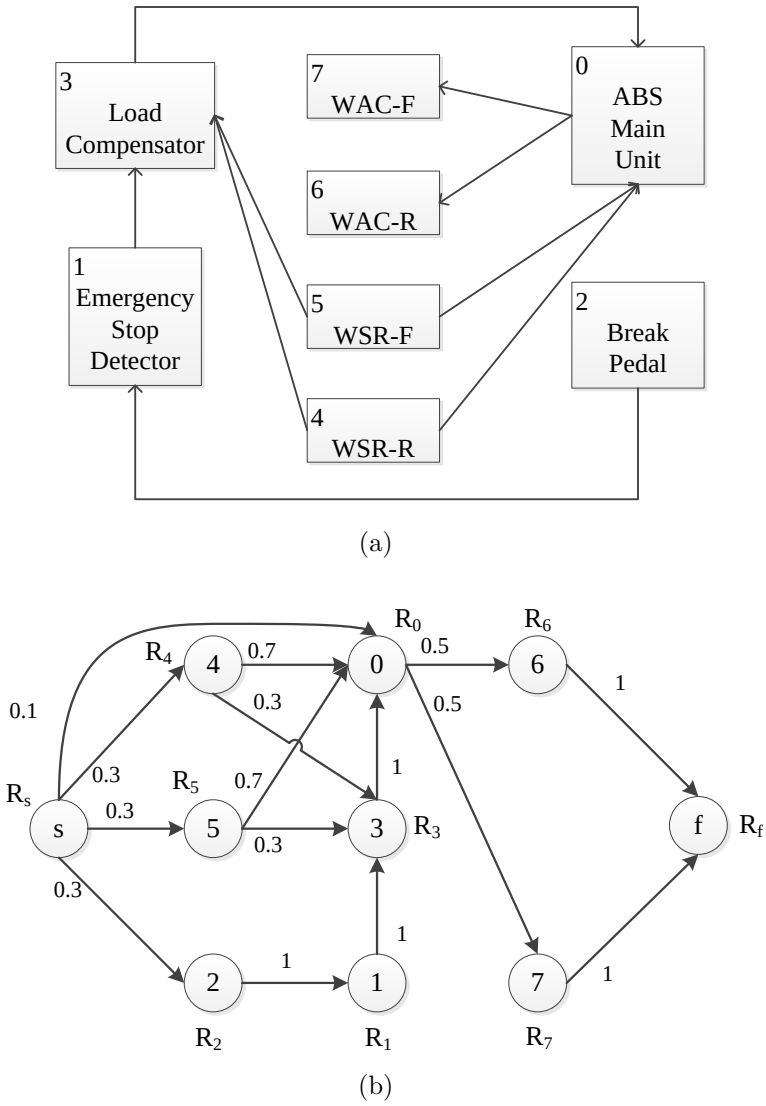


Figure 6.1: (a) Block diagram of the ABS testcase. (b) The DTMC model with states C and F removed.

tens and even hundreds of heterogeneous processing elements, including CPUs, DSPs, FPGAs, ASICs, and mixed digital/analog blocks for communication. In our assumed architecture platform (see Fig. 4.3), we assume two types of CPUs similar to the recent multicore proposals that integrate high-performance "big" and energy efficient "little" cores [61; 62; 63]. As HW components, we assume also two different types of FPGAs; one type that is slower but consumes less power and the other type that is faster but consumes more power. The FPGAs are assumed

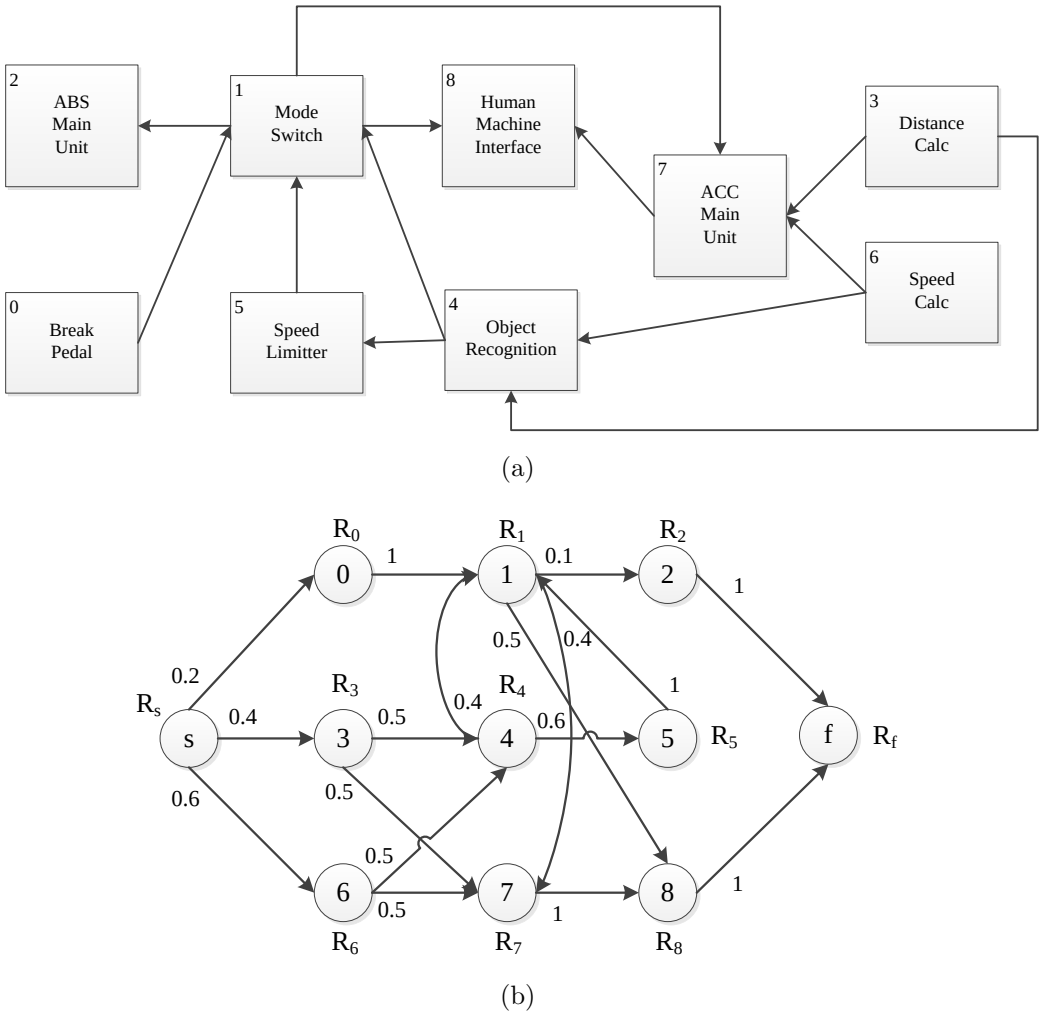


Figure 6.2: (a) Block diagram of the ACC testcase. (b) The DTMC model with states C and F removed.

to be faster than the CPUs and can offer increased parallelism; they may not be as fast as the ASIC cores, but, have the flexibility of reconfiguration.

Because we do not have available characterization data of actual execution times on different components, we adopt generic execution times and failure rates similar to [23], but with the assumption that execution times on HW components are shorter than those of SW components. For the power consumption of different components, we adopt values similar to those reported in [64]. Please note that, the generality of our tool is not affected by these assumptions because once we

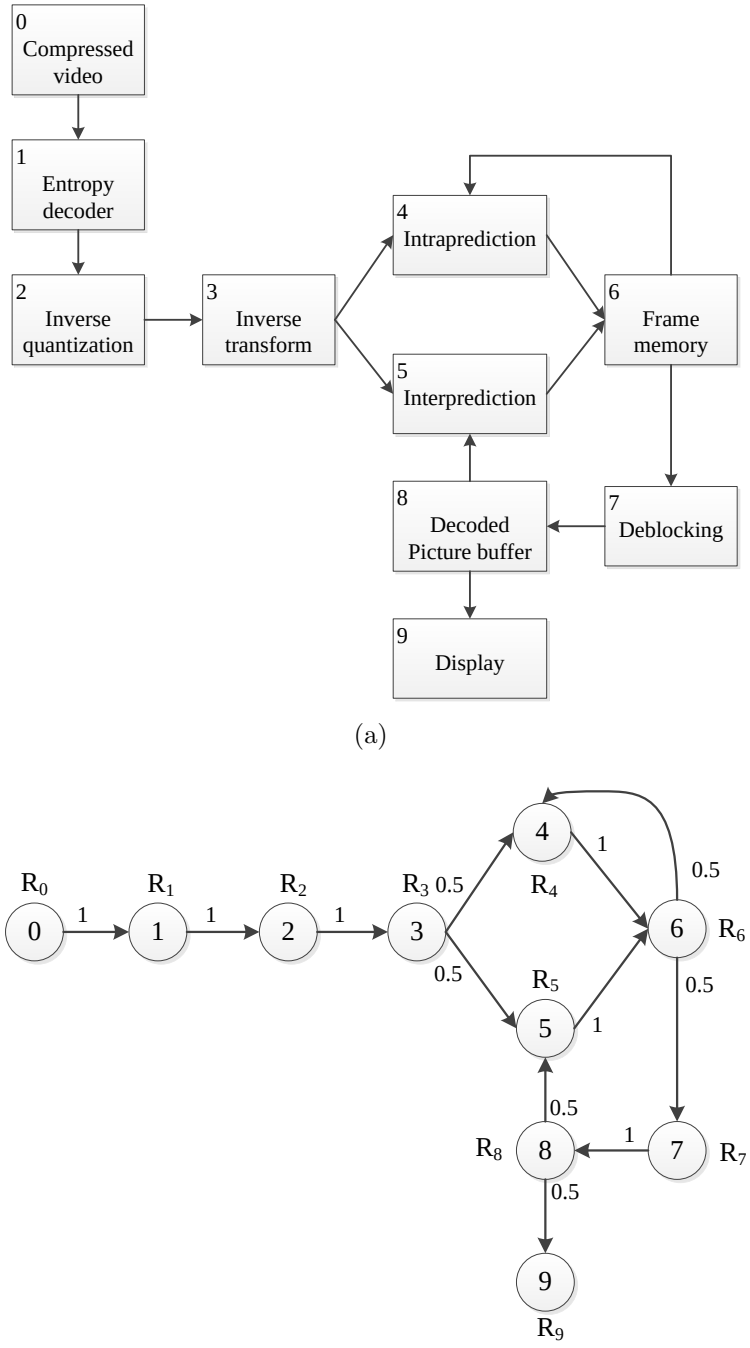


Figure 6.3: (a) Block diagram of the H.264 testcase. (b) The DTMC model with states C and F removed.

are given any HW/SW partitioning solution and more technology-specific data, our tool can find the best (as a compromise between reliability, performance, and energy) mapping solutions that are robust to pre-specified levels of uncertainty.

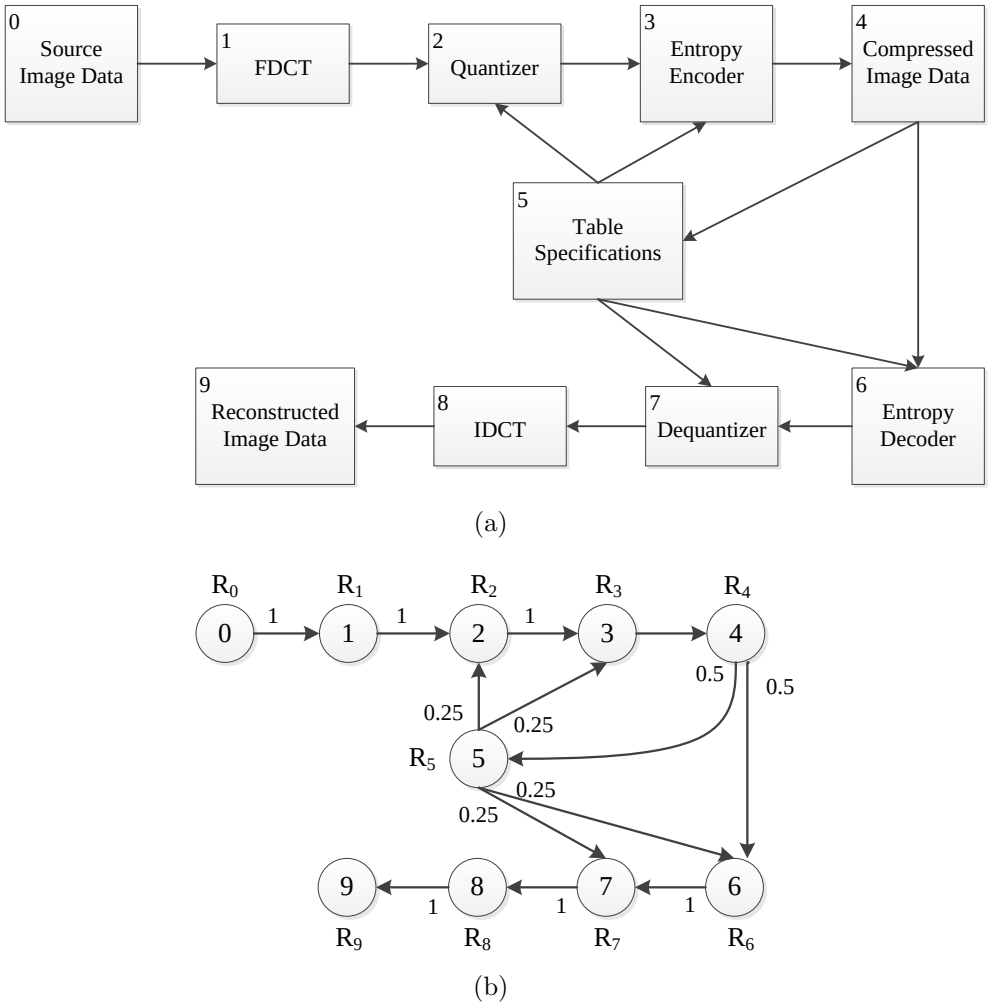
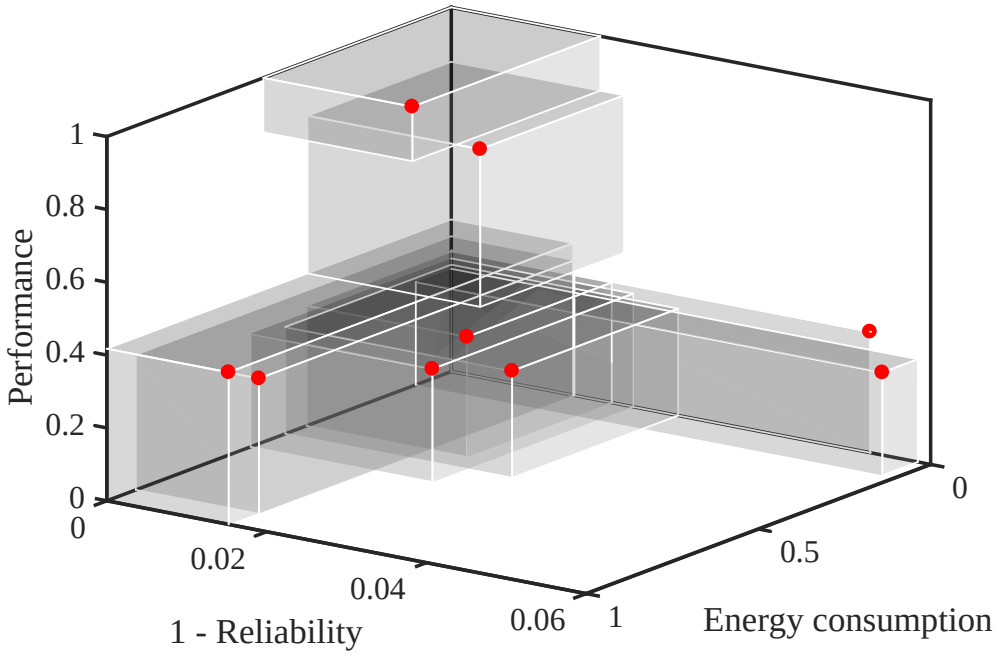


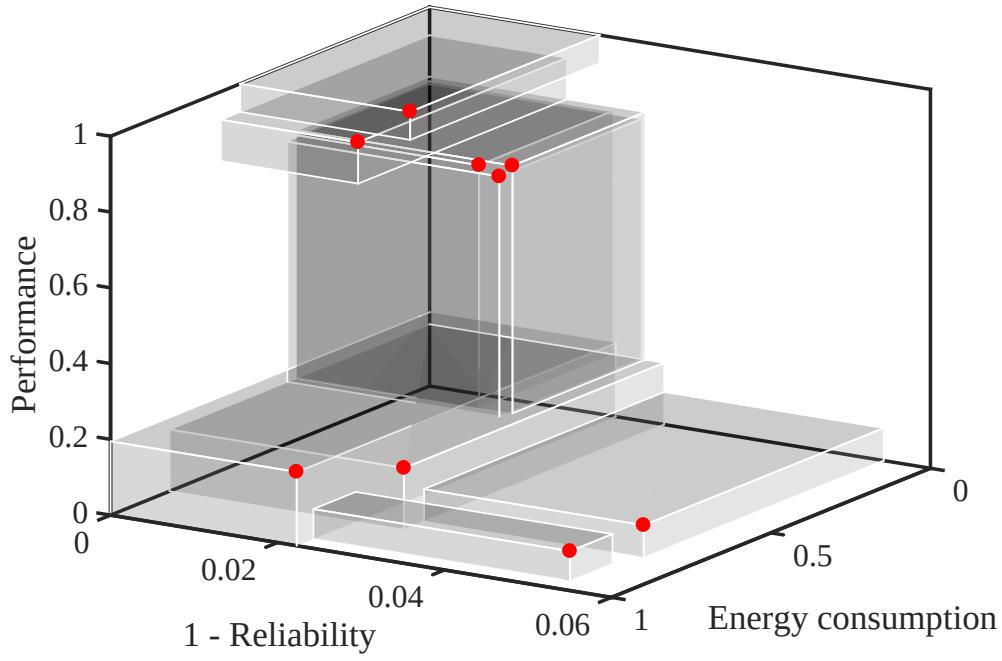
Figure 6.4: (a) Block diagram of the JPEG testcase. (b) The DTMC model with states C and F removed.

6.1.2 Pareto Frontiers

In the first set of simulations, we use our tool to identify the robust Pareto frontier in the *(1-reliability) vs. performance vs. energy* objective space for each of the testcases. All attributes are assumed to be affected by uncertainties, and therefore, they are estimated using the Monte Carlo technique described in Chapter 4. In order to generate Pareto frontiers that are scale independent, we normalize the performance and energy cost functions such that all values are inside the range $[0, 1]$. The normalization of a given cost function is done according to: $f_{norm} =$



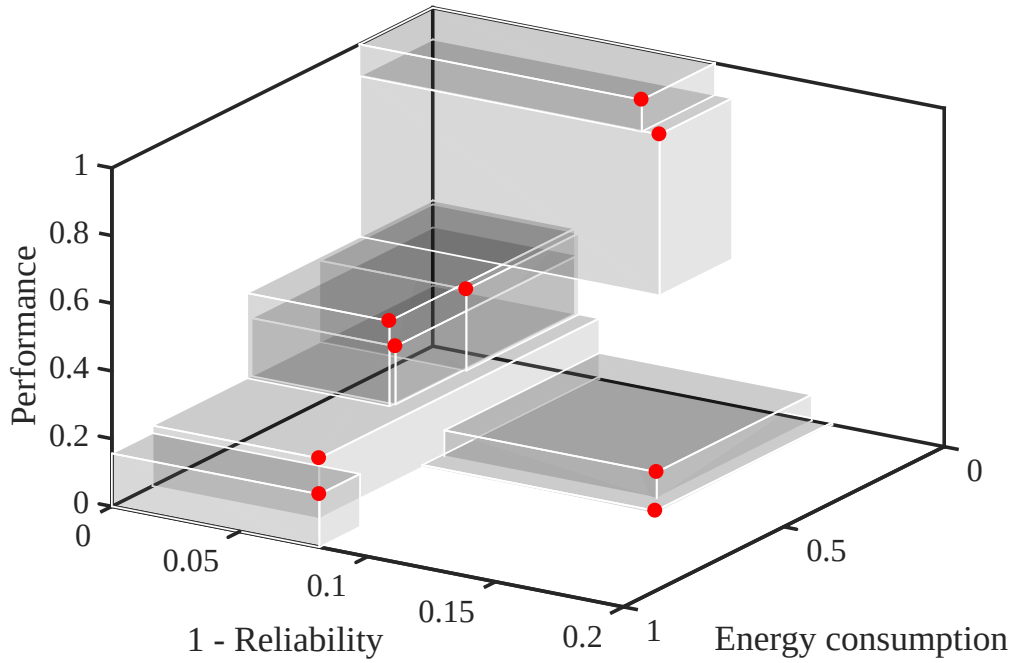
(a)



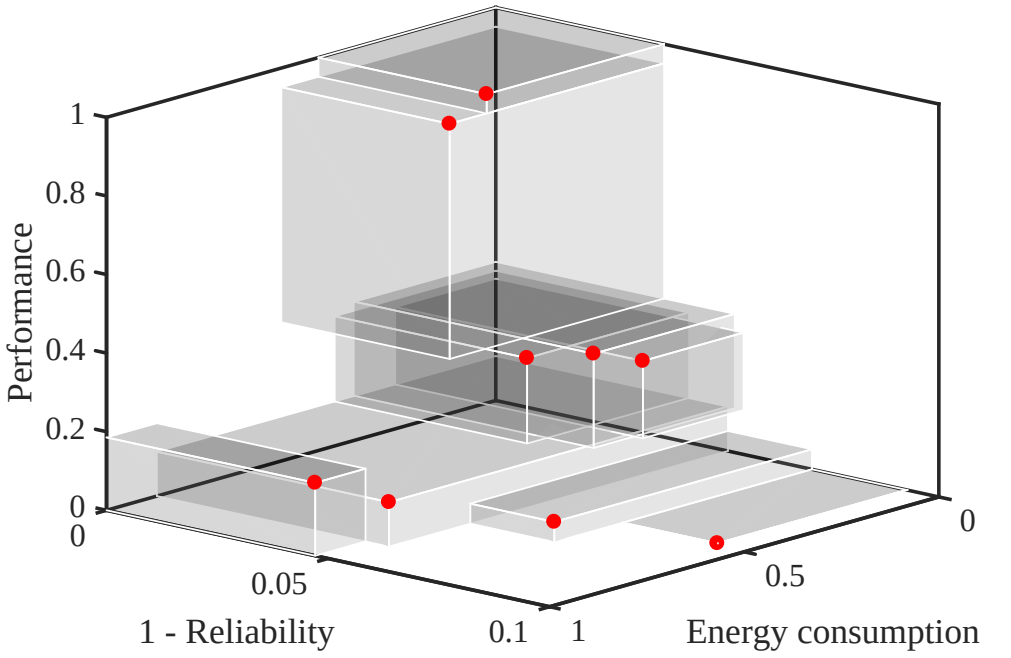
(b)

Figure 6.5: Robust Pareto frontiers of the simulated testcases for 5% injected uncertainty: (a) ABS, (b) ACC.

$(f - f_{min})/(f_{max} - f_{min})$, where f_{min} and f_{max} are the minimum and maximum or worst case scenario values of the respective objective cost function f. The cost



(a)



(b)

Figure 6.6: Robust Pareto frontiers of the simulated testcases for 5% injected uncertainty: (a) H.264, and (b) JPEG.

function *(1-reliability)* is already with values in the $[0, 1]$ range, hence, it does not require normalization.

The simplified Pareto frontiers of all the testcases are shown in Fig. 6.5 and 6.6 for a level of 5% injected uncertainty. These Pareto frontiers are simplified in the sense that they do not show all the actual solution points that were found to be on these frontiers during the execution of our tool. During this simplification, we basically select nine solution points from an actual Pareto frontier: three solution points that are as close as possible to the center of coordinates, and three pairs of two solution points that are very good in terms of only one of the three costs. We do this in order to keep these figures simple, yet to give the user enough solutions to choose from (the number of solutions can be changed if the user desired).

The solution points that are the closest to the system of coordinates represent solutions that the tool reports as being the best compromise among all three objectives. The other solution points can be selected if any of the three objectives is very important, depending on the application at hand. For example, if execution time or performance is highly critical, one of the two solution points that were found to offer very good performance (but with worse energy consumption and worse reliability) can be selected.

The ability to generate these 3D Pareto frontiers comprised of robust solution points (robust in the sense described in Chapter 4) represents one of the main contributions of this thesis.

6.1.3 Different Levels of Uncertainty

Second, we investigate how the Pareto frontiers change for different levels of injected uncertainty. Being able to study different levels of uncertainty can help in

two different scenarios. First, it can help us conduct *what if* type of investigations. For example, let us say that for a given technology node the uncertainty level is assumed to be 5%, but, that this value itself is not completely certain. In this case, we could investigate how the selected best solution found by the tool for uncertainty 5% would change if the assumed uncertainty level itself is varied. Such an investigation would help us see how the solution point moves in the 3D space and whether it still satisfies the desired figures of merit for the application at hand. This scenario is what we focus in this section.

However, as a second scenario, increasing levels of injected uncertainty can emulate the less deterministic design parameter values due to increased variations as we move toward deeper nonometer technology nodes. When the uncertainty increases, the availability of a tool like ours becomes even more important because it can help the designer to identify robust solution points, which are more immune against parameter spread. However, when migrating from a technology node to the next, the mean values (discussed in Chapter 4) of all the design parameters assumed to be affected by uncertainty must be scaled accordingly.

The different levels of uncertainty that we simulated are: 0% (no uncertainty, this is the deterministic case), 1%, 5%, and 10%. The simulations help to shed light over how the Pareto frontier changes with the change in the amount of injected uncertainty; under the assumption that the architecture platform remains the same in terms of number of components and floorplan. For brevity, we show here only the plots for the first testcase; the other testcases have similar plots. These Pareto frontiers are shown in Fig. 6.7.

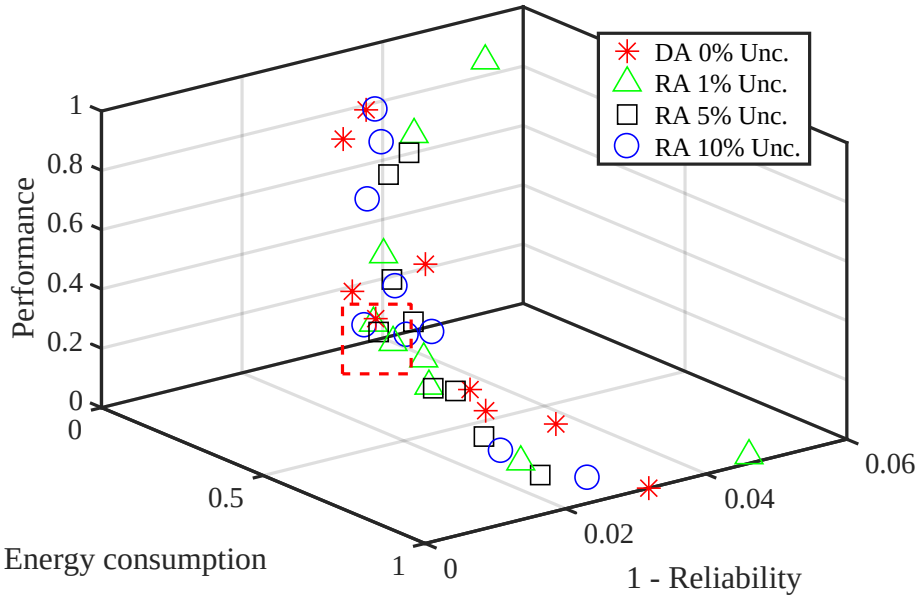


Figure 6.7: Pareto frontiers of the ABS testcase for different levels of injected uncertainty: 0%, 1%, 5%, and 10%.

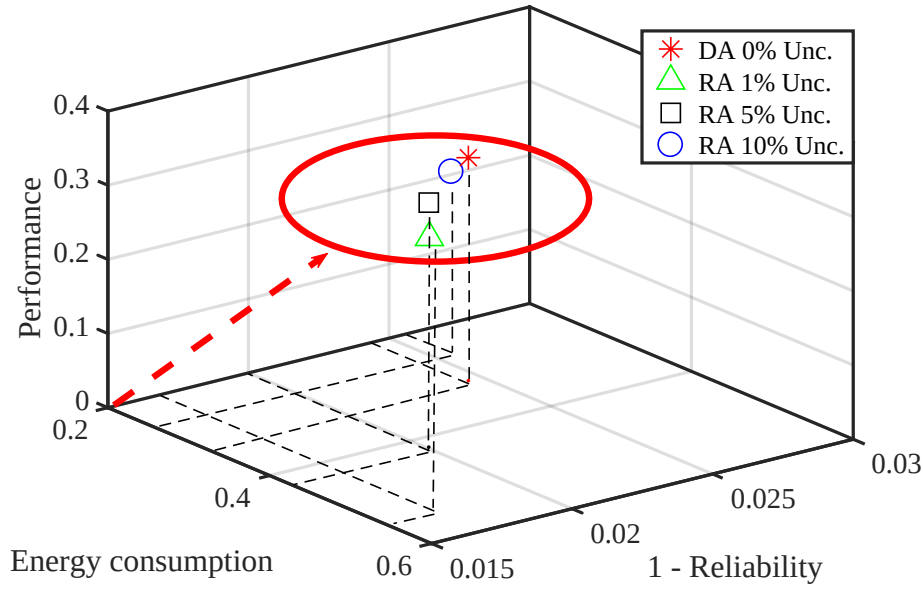


Figure 6.8: Comparison between the deterministic approach (DA) and the robust approach (RA).

Having the deterministic case as a reference, when uncertainty is injected, the previously deterministic and fixed parameter values are replaced with samples generated using various probability distributions, each characterized by a certain

mean and standard deviation pair. The standard deviation value that is used is directly related to the amount of desired uncertainty to be injected as discussed earlier in the thesis. Thus, a previously deterministic design solution point degenerates into a probability distribution, whose 95 percentile estimate represents the robust solution point that we use for constructing the robust Pareto frontiers. The location of this point is most likely different from the location of the previously deterministic design solution point. The amount of this change is within a vicinity whose size is dictated by the amount of uncertainty injected.

For example, this can be seen in the zoom-in picture from Fig. 6.8, which shows the four solution points for each of the four levels of injected uncertainty for a given mapping solution. The zero injected uncertainty represents the deterministic approach. Essentially, this figure illustrates how a solution point found by traditional deterministic approaches can be off from the robust design solution point identified by our tool for a given level of injected uncertainty. However, by using our tool, we can identify this shift and quantify each of the found solution points in terms of reliability, performance, and energy per assumed amount of uncertainty. Therefore, such a tool can aid embedded designers in finding the appropriate solution points to be selected for a given application domain. **Our tool provides the means to investigate these changes. We view this as another important contribution of this thesis.**

It is noteworthy to observe also that these types of investigations could be thought of also within a framework where we would be interested in for example altering the architecture platform and possibly the mapping solution between

different technology nodes or generations (characterized by different levels of uncertainty) such that the increased uncertainties affect minimally the quality of a previous design, while also having the desired confidence level that the estimated figures of merit in simulations are reflective of what they will be in reality when the entire system is manufactured and realized in practice. Such other types of investigations will be the subject of our future work.

6.1.4 Consideration of Both Different Levels of Uncertainty and Uncertainty Correlations

After conducting the comparison experiments for different levels of uncertainty (DLOU), we further investigate how the Pareto frontiers change when correlations between uncertainty sources are considered. The motivation example in Chapter 3 answered the question of why we need to consider different levels of uncertainty as well as uncertainty correlations. Then, in Chapter 4 we proposed a novel uncertainty-aware analysis technique which considers both different levels of uncertainty and uncertainty correlations (DLOU-UC). In this section, the proposed uncertainty-aware analysis technique is implemented as a framework tool called DESUU-II. We conduct comparison experiments for the Pareto frontiers obtained by the DLOU technique (DESUU tool) and the DLOU-UC technique (DESUU-II tool).

We assume all the CPUs in the target architecture are affected by the same uncertainty source with injected 10% level of uncertainty, and all the FPGAs in the target architecture are influenced by another same uncertainty source with injected

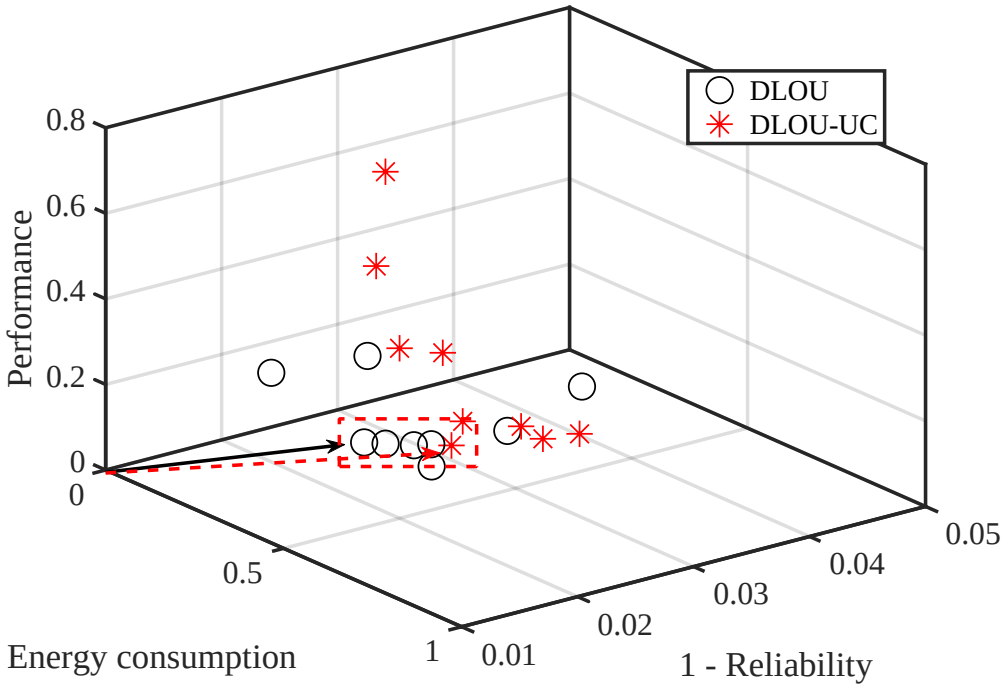


Figure 6.9: Pareto frontiers of the ABS testcase for DLOU and DLOU-UC.

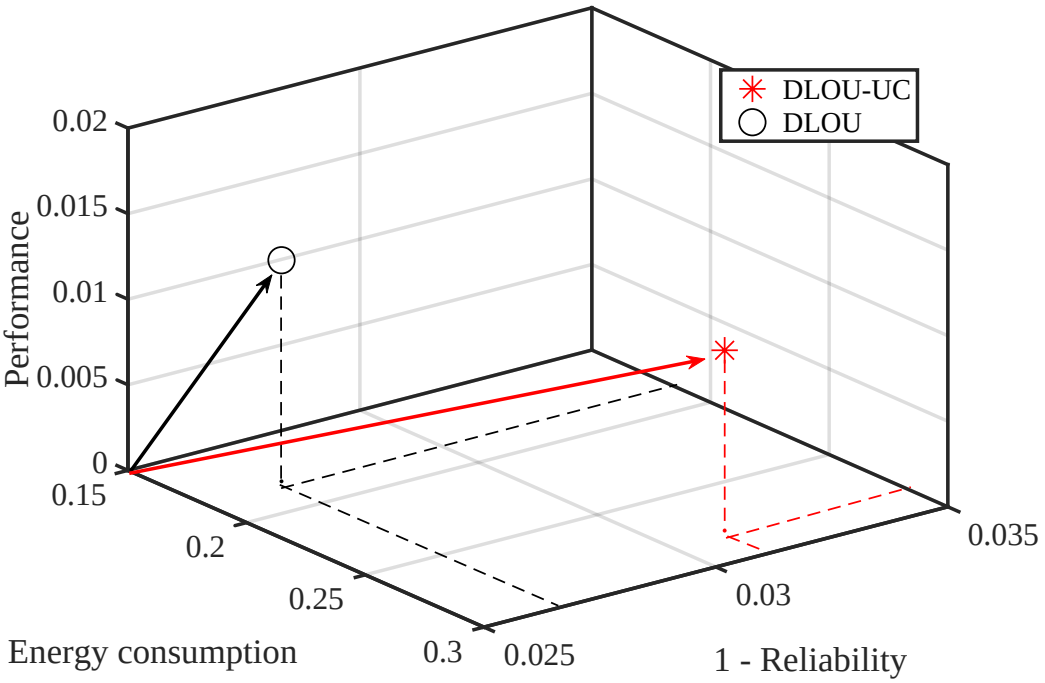


Figure 6.10: Comparison between the optimal solutions obtained by the DLOU technique and the DLOU-UC technique.

5% level of uncertainty. Therefore, there are two correlation groups according to the discussion in Chapter 4. We use the ABS testcase in these experiments.

We compare the Pareto frontiers obtained by the DLOU (DESUU tool) and the DLOU-UC (DESUU-II tool), the results are shown in Fig. 6.9. This figure shows that the Pareto frontier obtained in the DLOU-UC technique is located farther away from the system of coordinates when compared to the one obtained in the DLOU technique. This difference between the Pareto frontiers is expected because the DLOU technique analyzes the uncertainty among different components independently, while the DLOU-UC technique considers the uncertainty correlations among those components, which restricts the process of sampling of parameters for all components in the Monte Carlo simulations.

With different Pareto frontiers obtained, the locations of the optimal solutions obtained by the DLOU and the DLOU-UC are most likely to be different. For example, this can be observed in the zoom-in picture from Fig. 6.10, which shows that the optimal solution point obtained in the DLOU-UC technique is shifted when compared with the one obtained in the DLOU technique. In other words, if we use DLOU in uncertainty analysis, the solution may become suboptimal because we would miss taking into consideration of uncertainty correlations.

6.1.5 Scalability of the Computational Runtime and Convergence

Here, we look into the computational complexity and seek insights into the convergence of the algorithms. The computational complexity of the proposed tool is primarily affected by two factors, for a given testcase size. These factors are the number of iterations of the outer and inner loops from Fig. 4.1. To study the scalability of the computational runtime with the number of iterations of the

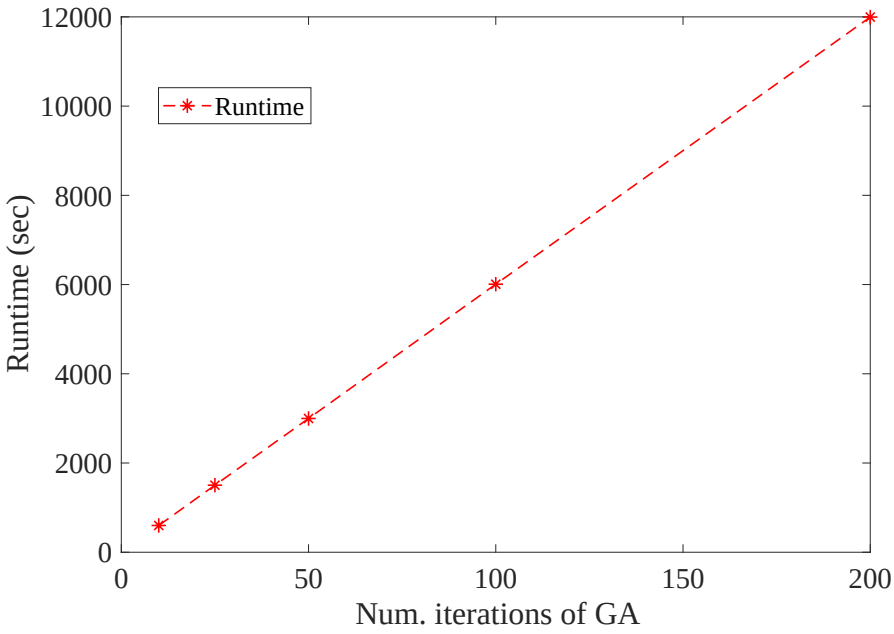


Figure 6.11: Computational runtime of our tool versus the number of iterations of the NSGA-II genetic algorithm.

outer loop, which corresponds to different number of solution populations explored by the genetic algorithm, we plot in Fig. 6.11 the computational runtime of our tool versus the number of iterations of the outer loop. This plot shows that the computational runtime scales linearly.

Inside each iteration of the outer loop, we have a number of iterations of the inner loop, which corresponds to the number of MC runs done for the purpose of attribute estimation under uncertainties. The computational runtime of our tool for a *single* iteration of the outer loop versus the number of MC runs is shown in Fig. 6.12. This plot shows again a linear dependency.

While the computational runtime is fairly reasonable for the size of the studied testcases, the question that arises though is how many iterations of the Monte Carlo algorithm should be used. In other words, we are interested in finding

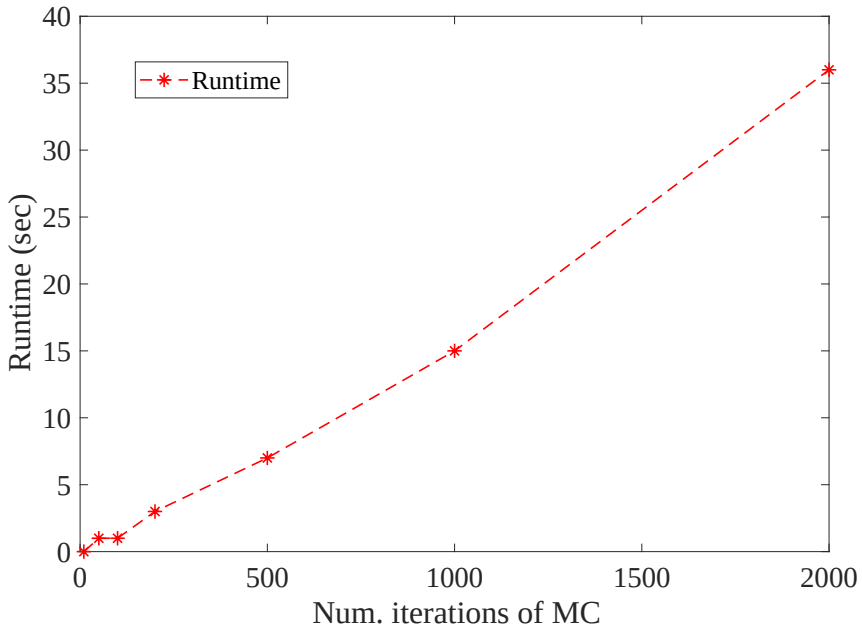
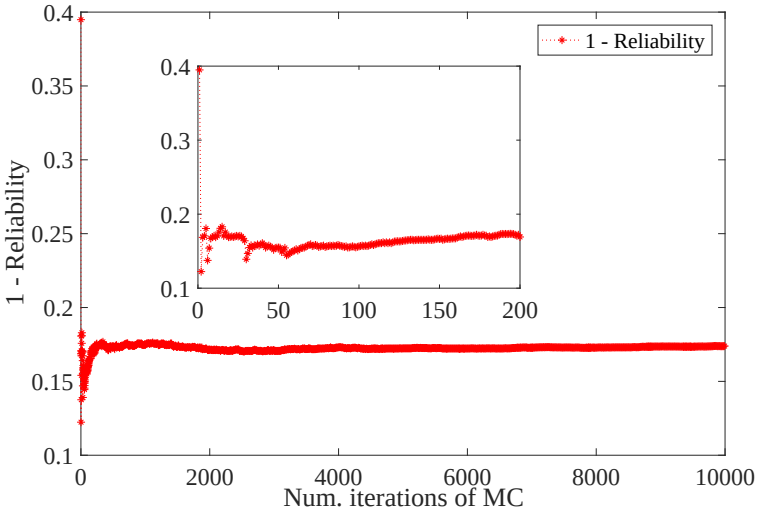


Figure 6.12: Computational runtime of only one iteration of the top-level outer loop versus the number of runs inside the Monte Carlo simulation.

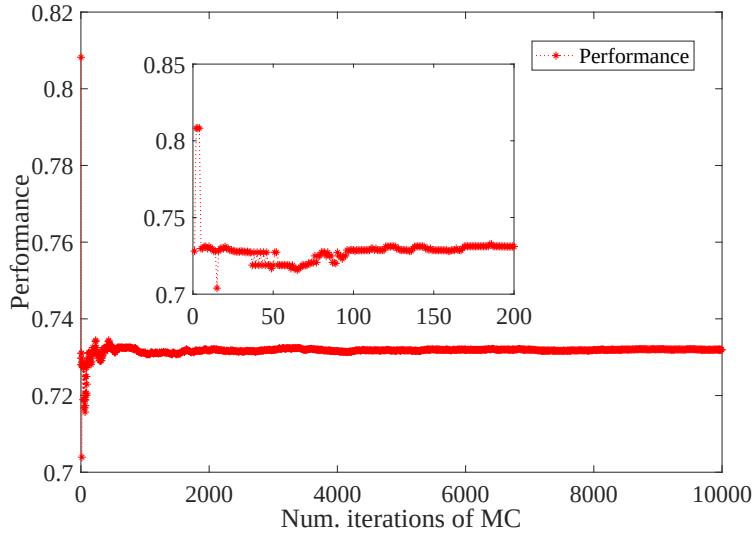
out what is the minimum number of MC runs after which convergence in the process of estimation is achieved. To answer this question, we looked at how the number of MC runs impacted the convergence of the estimation of the objective cost functions. This is illustrated by the plots in Fig. 6.13, where we can see that after about 2000 iterations of the MC algorithm, each of the three attributes does not fluctuate anymore.

6.2 Conclusion

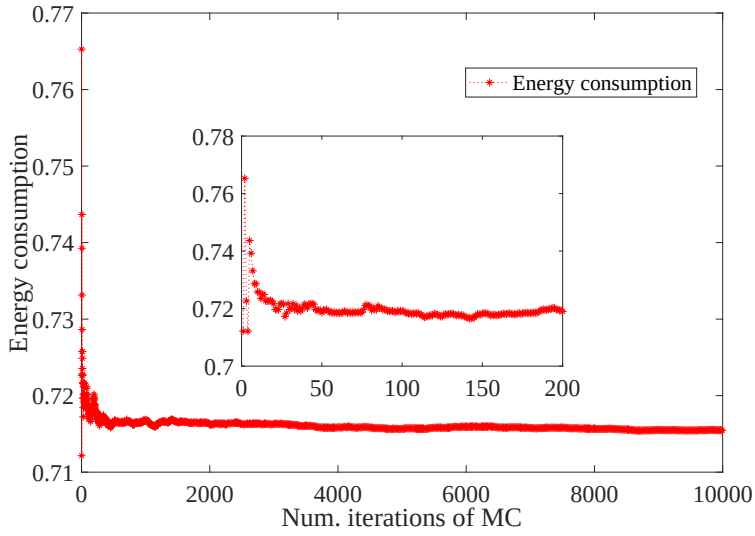
In this chapter, we report the Pareto frontiers obtained by the proposed design method, which we implemented as a tool called DESUU. In addition, we investigate different levels of injected uncertainty and provide simulation results. Furthermore, we propose a novel uncertainty-aware analysis technique, which considers both different levels of uncertainty and uncertainty correlations. The proposed uncertainty-aware analysis technique is implemented as a tool called DESUU-II. Last but not the least, we analyze the scalability of the computational runtime and convergence of the proposed design method.



(a)



(b)



(c)

Figure 6.13: Illustration of the convergence of the MC simulation based estimation.

CHAPTER 7

Conclusions

7.1 Conclusions and Future Work

To address the increased levels of design uncertainties in current and future embedded systems, we presented a design methodology for the design of embedded systems under uncertainties. We first formulate the problem of uncertainty aware mapping for multicore embedded systems platforms as a multi-objective optimization problem. Then, we present a solution to this problem that integrates uncertainty models and optimization algorithms constructed with Monte Carlo and evolutionary algorithms. The proposed methodology is implemented as a tool called DESUU that is capable of finding the robust Pareto frontiers in the objective space for a given testcase application, architecture platform, and given levels of injected uncertainties. Furthermore, to model uncertainty correlations between architecture components affected by multiple uncertainty sources, we propose a novel uncertainty-aware analysis technique with consideration of both uncertainty correlations and different levels of uncertainty. The proposed uncertainty-aware analysis technique is implemented as a tool called DESUU-II.

We conduct two sets of simulation experiments with different architecture platforms and testcases: comparison to traditional method and simulations on the proposed robust method, scalability analysis. Simulation results demonstrated the effectiveness of the proposed design method.

In future work, we plan to also include scheduling [65] into our problem formulation and to investigate architecture models that use networks-on-chip for communication. Architecture platform synthesis with direct consideration of all three objectives is also an interesting problem to investigate.

References

[1] M. Mahalingam and N. Ranganathan, "Timing-based placement considering uncertainty due to process variations," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 6, pp. 1007–1011, 2010.

[2] S.S. Sapatnekar, "Overcoming variations in nanometer-scale technologies," *IEEE J. on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 1, no. 1, pp. 5–18, 2011.

[3] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[4] A. DeHon, H.M. Quinn, and N.P. Carter, "Vision for cross-layer optimization to address the dual challenges of energy and reliability," *ACM/IEEE Design Automation and Test in Europe Conf. (DATE)*, 2010.

[5] S. Borkar, "Thousand core chips: a technology perspective," *ACM/IEEE Design Automation Conf. (DAC)*, 2007.

[6] A. Sangiovanni-Vincentelli, D. Densmore, J. Cong, and R. Marculescu, "System-level synthesis - functions, architectures, and communications," *ACM/IEEE Asia and South Pacific Design Automation Conf. (ASP-DAC) Tutorial*, 2008.

[7] S. Herbert, S. Garg, and D. Marculescu, "Exploiting process variability in voltage/frequency control," *IEEE Trans. on Very Large Scale Integr. Syst.*, vol. 20, no. 8, pp. 1392–1404, 2012.

[8] M. Lombardi, M. Milano, and L. Benini, "Robust scheduling of task graphs under execution time uncertainty," *IEEE Trans. on Computers*, vol. 62, no. 1, pp. 98–111, 2013.

[9] S. Maiti, N. Kapadia, and S. Pasricha, "Process variation aware dynamic power management in multicore systems with extended range voltage/frequency scaling," *IEEE Int. Midwest Symposium on Circuits and Systems (MWSCAS)*, 2015.

[10] C. U. Smith, G. A. Frank, and J. L. Cuadrado, "An architecture design and assessment system for software hardware codesign," *ACM/IEEE Design Automation Conference (DAC)*, 1985.

[11] S. Prakash and A. C. Parker, "Sos: Synthesis of application-specific heterogeneous multiprocessor systems," *Journal of Parallel and Distributed Computing*, vol. 16, no. 4, pp. 338–351, 1992.

[12] K. Keutzer, A. R. Newton, J. M. Rabaey, and A. Sangiovanni Vincentelli, "System-level design: Orthogonalization of concerns and platform-based design," *IEEE Trans. Comp. Aided Des. Integ. Cir. Sys.*, vol. 19, no. 12, pp. 1523–1543, 2006.

[13] A. Gerstlauer, C. Haubelt, A.D. Pimentel, T. P. Stefanov, D. D. Gajski, and J. Teich, "Electronic system-level synthesis methodologies," *IEEE Trans. Comp. Aided Des. Integ. Cir. Sys.*, vol. 28, no. 10, pp. 1517–1530, 2009.

[14] J. Teich, "Hardware/software co-design: Past, present, and predicting the future," *Proceedings of the IEEE*, vol. 100, no. 5, pp. 1411–1430, 2012.

[15] G. Martin, "Overview of the mpsoc design challenge," *ACM the 43rd Annual Design Automation Conference (DAC)*, 2006.

[16] J. Keinert, M. Streubhr, T. Schlichter, J. Falk, J. Gladigau, C. Haubelt, J. Teich, and M. Meredith, "Systemcodesigneran automatic esl synthesis approach by design space exploration and behavioral synthesis for streaming applications," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 1, pp. 1:1–1:23, 2009.

[17] J. Teich, T. Blickle, and L. Thiele, "An evolutionary approach to system-level synthesis," *IEEE International Workshop on Hardware/Software Co-Design*, 1997.

[18] L. Painton and J. Campbell, "Genetic algorithms in optimization of system reliability," *IEEE Transactions on Reliability*, vol. 44, no. 2, pp. 172–178, 1995.

[19] B. P. Dave and N. K. Jha, "Cohra: Hardware-software co-synthesis of hierarchical distributed embedded system architectures," *IEEE International Conference on VLSI Design*, 1998.

[20] Y. Ren and J. Bechta Dugan, "Design of reliable systems using static and dynamic fault trees," *IEEE Transactions on Reliability*, vol. 47, no. 3, pp. 234–244, 1998.

[21] M. Forster and M. Trapp, "Fault tree analysis of software-controlled component systems based on second-order probabilities," *IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2009.

[22] L. Grunske, B. Kaiser, and Y. Papadopoulos, "Model-driven safety evaluation with state-event-based component failure annotations," *International Symposium on Component-Based Software Engineering (CBSE)*, 2005.

[23] I. Meedeniya, A. Aleti, and L. Grunske, "Architectur-driven reliability optimization with uncertain model parameters," *The Hournal of Systems and Software*, vol. 85, no. 10, pp. 2340–2355, 2012.

[24] F. Khosravi, M. Muller, M. Glaß, and J. Teich, "Uncertainty-aware reliability analysis and optimization," *ACM/IEEE Design, Automation and Test in Europe Conference (DATE)*, 2015.

[25] C. Erbas, S. Cerav-Erbas, and A.D. Pimentel, "Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design," *IEEE. Trans. on Evolutionary Computation*, vol. 10, no. 3, pp. 358–374, 2006.

[26] K. Sigdel, C. Galuzzi, K. Bertels, M. Thompson, and A.D. Pimentel, "Evaluation of runtime task mapping using the sesame framework," *Int. J. Reconfig. Comp.*, vol. 12, pp. 1–17, 2012.

[27] S.H. Kang, H. Yang, L. Schor, I. Bacivarov, S. Ha, and L. Thiele, "Multi-objective mapping optimization via problem decomposition for many-core

systems," *ESTImedia*, 2012.

[28] W. Quan and A.D. Pimentel, "A hybrid task mapping algorithm for heterogeneous mpsocs," *ACM Trans. on Embedded Computing Systems*, vol. 14, no. 1, pp. 14, 2015.

[29] G.Ascia, V. Catania, and M.Palesi, "A ga-based design space exploration framework for parameterized system-on-a-chip platforms," *IEEE Trans. on Evolutionary Computation*, vol. 8, no. 4, pp. 329–346, 2004.

[30] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A.S. Vincentelli, "Metropolis: an integrated electronic system design environment," *IEEE Computer*, vol. 36, no. 4, pp. 45–52, 2003.

[31] A. Cassidy, J. Paul, and D. Thomas, "Layered, multi-threaded, high-level performance design," *ACM/IEEE Int. Conf. Design, Automation and Test in Europe (DATE)*, 2003.

[32] R. Domer, A. Gerstlauer, J. Peng, D. Shin, L. Cai, H. Yu, S. Abdi, , and D.D. Gajski, "System-on-chip environment: a specc-based framework for heterogeneous mpsoc design," *EURASIP J. on Embedded Systems (JES)*, vol. 2008, no. 3, pp. 1–13, 2008.

[33] A.D. Pimentel, "The artemis workbench for system-level performance evaluation of embedded systems," *J. of Systems Architecture*, vol. 3, no. 3, pp. 181–196, 2008.

[34] H. Nikolov, T. Stefanov, and E. Deprettere, "Systematic and automated multiprocessor system design, programming, and implementation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 3, pp. 542–555, 2008.

[35] W. Quan and A.D. Pimentel, "A hierarchical run-time adaptive resource allocation framework for large-scale mpsoc systems," *Design Automation for Embedded Systems*, vol. 20, no. 4, pp. 311–339, 2016.

[36] A. Aleti, B. Guhnova, L. Grunske amd A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *IEEE Trans. on Software Engineering*, vol. 39, no. 5, pp. 658–683, 2013.

[37] I. Meedeniya, I. Moser, A. Aleti, and L. Grunske, "Architecture-based reliability evaluation under uncertainty," *QoSA/ISARCS*, 2011.

[38] I. Meedeniya, I. Moser, A. Aleti, and L. Grunske, "Evaluating probabilistic models with uncertain model parameters," *Softw. Syst. Model*, vol. 13, no. 4, pp. 1395–1415, 2014.

[39] F. Khosravi, H. Aliee, and J. Teich, "System-Level Reliability Analysis Considering Imperfect Fault Coverage," *15th IEEE/ACM Symposium on Embedded Systems for Real-Time Multimedia (ESTIMedia)*, 2017.

[40] C. Bolchini, M. Carminati, A. Miele, A. Das, A. Kumar, and B. Veeravalli, "Run-time mapping for reliable many-cores based on energy/performance trade-offs," *IEEE Int. Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2013.

[41] A.Y. Yamamoto and C. Ababei, "Unified reliability estimation and management of noc based chip multiprocessors," *Microprocessors and Microsystems*, vol. 38, no. 1, pp. 53–63, 2014.

[42] M.G. Moghaddam, A.Y. Yamamoto, and C. Ababei, "Investigation of dvfs based dynamic reliability management for chip multiprocessors," *IEEE Int. Conf. High Performance Computing and Simulation (HPCS)*, 2015.

[43] T. Gillespie, *Fundamentals of Vehicle Dynamics*, PA: SAE: Warrendale, 1992.

[44] J. Y. Wong, *Theory of Ground Vehicles*, New York: Wiley, 1993.

[45] K. Kogut, "Anti-lock braking system modelling and parameters identification," *IEEE International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2014.

[46] L. Grunske, "Towards an integration of standard component-based safety evaluation techniques with saveccm," *Quality of Software Architectures (QoSA)*, 2006.

[47] W.L. Oberkampf, S.M. Deland, B.M. Rutherford, K.V. Diegert, and D.F. Alvin, "Estimation of total uncertainty in modeling and simulation," *SANDIA REPORT, SAND2000-0824*, 2000.

[48] Bilal M. Ayyub and George J. Klir, *Uncertainty Modeling and Analysis in Engineering and the Sciences*, Chapman and Hall/CRC, 2006.

[49] Y. Li, J. Chen, , and L. Feng, "Dealing with uncertainty: a survey of theories and practices," *IEEE Trans. on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2463–2482, 2013.

[50] G. Kahn, "The semantics of a simple language for parallel programming," *IFIP Congress74*, 1974.

[51] B.P. Dave, G. Lakshminarayana, and N.K. Jha, "Cosyn: hardware-software cosynthesis of heterogeneous distributed embedded systems," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 92–104, 1999.

[52] M. Lopez Vallejo and J.C. Lopez, "On the hardware-software partitioning problem: system modeling and partitioning techniques," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 8, no. 3, pp. 269–297, 2003.

[53] A.D. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Trans. on Computers*, vol. 55, no. 2, pp. 99–112, 2006.

[54] Indika Meedeniya, *Architecture Optimisation of Embedded Systems under Uncertainty in Probabilistic Reliability Evaluation Model Parameters*, Faculty of Information and Communication Technologies, Swinburne University of Technology, 2012.

[55] R.C. Cheung, "A user-oriented software reliability model," *IEEE Transactions on Software Engineering*, vol. 6, no. 2, pp. 118–125, 1980.

[56] Cagkan Erbas, *System-Level Modeling and Design Space Exploration for Multiprocessor Embedded System-on-Chip Architectures*, Faculty of Science, Amsterdam University, 2006.

[57] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[58] "Kalyanmoy deb (2016) multi-objective nsga-ii code in c," http://www.egr.msu.edu/~kdeb/codes.shtml.

[59] W. Quan, *Scenario-based run-time adaptive Multi-Processor System-on-Chip*, Faculty of Science, Amsterdam University, 2015.

[60] J.R. Smith, "The h.264 video coding standard," *IEEE Computer Society*, vol. 13, 2006.

[61] G. Wallace, "The jpeg still picture compression standard," *IEEE Trans. on Consumer Electronics*, vol. 38, no. 1, 1992.

[62] "Arm big.little technology (2017)," https://developer.arm.com/technologies/big-little.

[63] N. Chitlur, G. Srinivasa, S. Hahn, P.K. Gupta, D. Reddy, D. Koufaty, P. Brett, A. Prabhakaran, L. Zhao, N. Ijih, S. Subhaschandra, S. Grover, X. Jiang, and R. Iyer, "Quickia: exploring heterogeneous architectures on real prototypes," *IEEE Int. Symposium on High Performance Computer Architecture (HPCA)*, 2012.

[64] X. Liang, M. Nguyen, and H. Che, "Wimpy or brawny cores: a throughput perspective," *Journal of Parallel and Distributed Computing archive*, vol. 73, no. 10, pp. 1351–1361, 2013.

[65] V. Konstantakos, A. Chatzigeorgious, S. Nikolaidis, and T. Laopoulos, "Energy consumption estimation in embedded systems," *IEEE Trans. on Instrumentation and Measurement*, vol. 57, no. 4, 2008.

[66] P. Pop, K.H. Poulsen, V. Izosimov, and P. Eles, "Scheduling and voltage scaling for energy and reliability trade-offs in fault-tolerant time-triggered embedded systems," *IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2007.

[67] A. Birolini, *Reliability Engineering: Theory and Practice*, Springer Verlag, 2010.