Reliability-Aware System Synthesis *

Michael Glaß, Martin Lukasiewycz, Thilo Streichert, Christian Haubelt, and Jürgen Teich
University of Erlangen-Nuremberg, Germany
{glass, martin.lukasiewycz, streichert, haubelt, teich}@cs.fau.de

Abstract

Increasing reliability is one of the most important design goals for current and future embedded systems. In this paper, we will put focus on the design phase in which reliability constitutes one of several competing design objectives. Existing approaches considered the simultaneous optimization of reliability with other objectives to be too extensive. Hence, they firstly design a system, secondly analyze the system for reliability and finally exchange critical parts or introduce redundancy in order to satisfy given reliability constraints or optimize reliability. Unfortunately, this may lead to suboptimal designs concerning other design objectives. Here, we will present a) a novel approach that considers reliability with all other design objectives simultaneously, b) an evaluation technique that is able to perform a quantitative analysis in reasonable time even for real-world applications, and c) experimental results showing the effectiveness of our approach.

1. Introduction

Integrated circuits applied in consumer products, prototyping systems, or applications from automotive and avionic industries have to fulfill different constraints like cost, performance, or reliability. Especially, electronic control units (ECUs) from automotive and avionic systems demand a high reliability while they are operating under different radiation and varying temperature conditions. While high temperature differences stresses electronics and accelerates aging of components, high radiation can lead to single event upsets (SEU). These SEUs are caused by charging of single transistors and results in a fault of the circuit. Whether this fault affects the circuit's functionality permanently or temporarily, depends on the device as well as on the location of the fault. Assume for example an SRAM-based FPGA which is configured with a certain functionality and is corrupted by a bit flip of a functional part. Depending on the location of the SEU, an error occurs which is either temporary or permanent. If the SEU occurs in the configuration memory cell, the functionality will be corrupted until the next reconfiguration permanently. Otherwise, the SEU affects only flip flops or inverter and leads to transient faults. The influence of SEUs to FPGAs has been intensively investigated in [10]. Interestingly, FPGA-based ECUs which are applied in the heavy-ion detector in Cern [1] have to operate under high radiation conditions. The FPGAs in these ECUs are periodically read-back and reconfigured, in order to guarantee a fault free operation. On the other hand, transient faults are prognosed to occur more often [16] in high-density ASICs. Due to shrinking transistor sizes, the switching capacity decreases and radiation is more likely to charge the gates.

Reliability engineering is a discipline that covers the whole life cycle of a system, from design time over production and test to operation. At all these life cycle phases, reliability engineering aims at decreasing the probability that a system fails as well as providing a service in case of a failure. Existing methodologies for designing reliable systems firstly design the system, secondly analyze the system for reliability and finally exchange critical parts or introduce redundancy in order to satisfy given reliability constraints or optimize reliability. Unfortunately, this may lead to suboptimal designs concerning design objectives considered before reliability analysis. Thus, this paper presents a novel system-level methodology for automatic synthesis of reliable integrated systems. Starting from a behavioral description in form of a data flow graph (DFG) and a resource graph for modeling all architectural alternatives, we are able to synthesize reliable data paths which are optimal with respect to multiple objectives. An illustrating example for the methodology presented in this paper is given in Fig. 1. In this example an architecture of a 4-tap FIR filter should be synthesized. The hardware implementation can consist of different types of adders and multipliers. Moreover, it can be chosen between different redundancy degrees. The area, latency, and reliability for different kinds of adders and multipliers as depicted in the following table has been taken from [15] [1]:

Resource Type	Area	Delay	Reliability
ADD1	1	2	0.999
ADD2	2	1	0.969
ADD3	4	1	0.987
MUL1	2	2	0.999
MUL2	4	1	0.969

With these different arithmetic components it is possible to compose lots of different architectures which are all rep-

*Supported in part by the German Science Foundation (DFG), SFB 694, SPP 1148

[1] Reliabilities of electrical components are listed also in [2].

978-3-9810801-2-4/DATE07 © 2007 EDAA

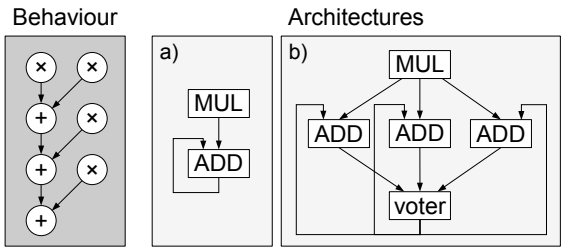

Figure 1. The behavior of a 4-tap FIR filter is modeled with a DFG. Different architectures a) without and b) with redundancy can be synthesized. To simplify matters, delay elements of the filter are neglected.

resenting a 4-tap FIR filter. However, let us consider the following three implementations:

- If the FIR filter is implemented using the architecture shown in Fig. 1a) and MUL1 as well as ADD2 are used for arithmetic operation, the reliability of the entire architecture is 0.906, the area consumption is 4 and the latency is 9.
- In order to optimize the latency, it is possible to replace MUL1 with MUL2. The resulting architecture has a reliability of 0.802, an area consumption of 6 and a latency of 5.
- For increasing the reliability of the first implementation, two possibilities exist: a) to exchange ADD2 with ADD1 or ADD3 and b) to make use of redundancy (see architecture b) in Fig. 1). If we consider architecture b) with MUL1 and ADD2, the reliability is 0.988, the area is 8 and the latency is 9.

All these implementations of an FIR filter differ in their reliability, area and latency. Until now it has been up to the designer to take the design decision about the best implementation. However, with our proposed multi-objective synthesis approach we help the designer for an unbiased decision making and allow for an automatic generation of reliable designs. Therefore, this paper contributes with a novel methodology having the following features:

- Selection of resources which are functional equivalent, but have different reliabilities as well as area and latency parameters.
- Simultaneous optimization of different objectives, like reliability, area, and latency.
- Resource sharing: Different functional nodes of a DFG may be bound and scheduled onto the same resource.

The rest of this paper is organized as follows: Section 2 discusses the prior work. Section 3 describes our novel design methodology including our approach to introduce redundancy and to analyze the reliability of the overall design. Section 4 presents our experimental data. The paper is concluded with a summary in Section 5.

2. Related Work

Up to now, several approaches have been presented for analyzing systems with respect to reliability and fault tolerance. An overview of these techniques can be found in [3]. In the last years, these techniques were integrated in the synthesis phase of a system in order to automatically synthesize reliable or fault-tolerant systems.

An early approach in the field of fault-tolerant ASIC design has been introduced by Karri et al. [9] who minimize the hardware overhead caused by replication. As input data for their synthesis, the authors' approach requires a data flow graph (DFG), a redundancy ratio and the latency. With this information different transformations are performed on the DFG which reduce the cost for redundancy. The results are impressive, since the savings for hardware costs range between 0% and 35.71% for a redundancy degree from 2 to 7 in their example.

Another approach [15, 16] tries to maximize reliability by selectively introducing redundancy for detecting soft errors. This approach is able to consider resources with the same functionality but different area, delay or reliability values. It first selects the most reliable resources needed to bind a certain DFG onto the resources and schedules it. If this design does not meet area or latency constraints, one resource is exchanged with a smaller or faster but less reliable resource. With this exchange strategy, the presented synthesis strategy tries to fulfill the constraints area and latency while maximizing the reliability. Note that only one objective (reliability) can be optimized while area and latency are constraints.

A co-design framework which assesses reliability properties on system level has been presented by Bolchini et al. [6] who propose a two step approach. In the first step the authors perform a partitioning respecting constraints like area, power, costs, etc. and in a second step the reliability of critical parts is increased. Unfortunately, this second step might worsen the solution obtained in the first step. Anyway, this approach is interesting since different methods are chosen in the second step for increasing reliability.

Coit and Smith [8] propose a relevant approach which synthesizes reliable systems with the help of Genetic Algorithms (GA). The GA selects appropriate resources and determines the degree of redundancy such that the cost is minimal for a given reliability. For estimating the reliability which is an objective function for the GA, the authors apply a neural network.

Similar to the last approach, our system synthesis performs a design space exploration with the help of Evolutionary Algorithms. But, we allow for multiple objectives

using state of the art Multi-Objective Evolutionary Algorithms [4]. Furthermore, the reliability of the synthesized system is determined based on Binary Decision Diagrams (BDDs). This calculation is a matter of the following section.

3. Integrating Reliability Optimization into System Synthesis

Until now, reliability was considered to be too extensive for an optimization with all other and often competing objectives simultaneously [6]. In this section, we will introduce our methodology to integrate reliability as an objective in multi-objective design space exploration. Starting with an introduction of our system model, we will briefly describe how our methodology selects resources and binds operations. In particular the binding has been extended to permit *multiple bindings* of operations which introduces redundancy into the systems. Afterwards, our efficient technique to determine the reliability of complex system structures using BDDs will complete our methodology description.

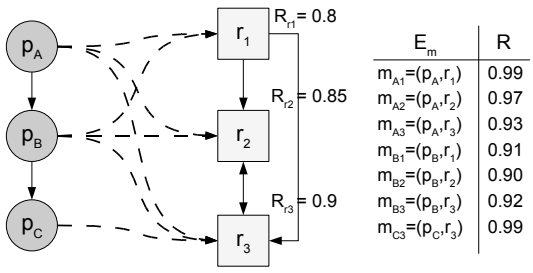

Figure 2. A specification graph consisting of a problem graph, an architecture graph and mapping edges.

Fig. 2 shows our system model represented through a so called *specification graph* $G_\mathrm{s} = (G_\mathrm{p}, G_\mathrm{a}, E_\mathrm{m})$ [5]. The acyclic *problem graph* $G_\mathrm{p} = (V_\mathrm{p}, E_\mathrm{p})$, shown on the left, models the behavior of the system and can in this case be seen as a DFG with operations $p \in V_\mathrm{p}$ and data dependencies $e \in E_\mathrm{p}$. The *architecture graph* $G_\mathrm{a} = (V_\mathrm{a}, E_\mathrm{a})$ on the right models resources $r \in V_\mathrm{a}$ and their interconnections $e \in E_\mathrm{a}$. The dashed edges are called *mapping edges* $m \in E_\mathrm{m}$ and indicate, that the operation in the problem graph can be implemented on the resource in the architecture graph. To model the reliability of the used resources, the reliability of each resource is annotated as an attribute in the architecture graph. To model the reliability of an operation computed on a specific resource, the reliability is annotated as an attribute on the mapping edge connecting the operation with this resource. Hence, this enables addressing defects and failures of a resource as well as logical failures of an operation being computed. Our approach is able to handle reliability attributes as probabilities as well as exponential or Weibull distribution functions.

The simultaneous optimization of reliability with other design objectives is performed by the design space exploration, which also performs the system synthesis. System synthesis consists of three main steps, which are 1.) *allocate* a set of resources, 2.) validly *bind* each operation to an allocated resource and 3.) determine a valid *schedule*. As existing design space exploration approaches [11, 13, 14] are limited such in a way that each operation has to be bound exactly once, they fail to generate highly reliable solutions through redundancy. Thus, we have extended our design space exploration [13] by

1. generating solutions with multiple operation instances
2. providing a reliability evaluation method for our system model.

Our design space exploration uses Multi-Objective Evolutionary Algorithms from [4]. While the allocation $\alpha \subseteq V_\mathrm{a}$ of resources is decoded from a bit vector where each bit encodes whether the associated resource is allocated or not, the binding $\beta \subseteq E_\mathrm{m}$ is decoded from priority lists. These priority lists enforce a decoding order about which function is bound first and on which resource. In order to generate more than one operation instance, for each operation an additional parameter is encoded in the chromosome indicating the number of mapping edges to be selected. These parameters also undergo genetic operations, i.e., mutation and crossover, in order to guide the search towards optimal designs.

3.1. Multiple Binding for Redundancy

The concept of *redundancy*, which is the existence of more than one means for performing a required function in an item [3], is well known for the ability to increase the reliability of a system. In common system synthesis models, each operation in the problem graph is bound to exactly one resource from the architecture graph. This lack of redundancy simplifies the reliability analysis of the system to a series structure of all components of the system. Consider the example in Fig. 2 with p_A being bound to r_1, p_B being bound to r_2 and p_3 being bound to r_3. The reliability of this example calculates to $R_{r1} \cdot R_{r2} \cdot R_{r3} \cdot R(m_{A1}(p_A, r_1)) \cdot R(m_{B2}(p_B, r_2)) \cdot R(m_{C3}(p_C, r_3)) = 0.546$.

Since our analysis technique is efficient enough to handle complex system structures, we introduce the ability to bind one operation to a set of resources, thus, creating several instances of one operation in the implementation of the system.

Our new definition of a feasible binding $\beta = \bigcup_{p \in V_\mathrm{p}} I_p$ is:

1. For each operation $p \in V_\mathrm{p}$ a set $I_p \subseteq E_\mathrm{m}$ of outgoing mapping edges is activated. Each mapping edge $m \in I_p$ represents one *operation instance* of operation p.

2. Each operation instance $m = (p, r) \in \beta$ ends at an allocated resource $r \in \alpha$.

3. For at least one set $L \subseteq \beta$ containing exactly one operation instance of each operation, the instances $m = (p, r)$ and $\widetilde{m} = (\widetilde{p}, \widetilde{r})$ in L of each data dependency $(p, \widetilde{p}) \in E_p$ have to be bound either to the same, or to two adjacent resources.

$$\begin{aligned} &\exists L : \forall (p, \widetilde{p}) \in E_{\mathrm{p}}, m = (p, r), \widetilde{m} = (\widetilde{p}, \widetilde{r}) \in L : \\ &r = \widetilde{r} \vee (r, \widetilde{r}) \in E_{\mathrm{a}} \\ &\text{with } L \subseteq \beta : \forall p \in V_{\mathrm{p}}\ \exists!\ m = (p, r) \in L \end{aligned} \tag{1}$$

The first requirement makes it possible to bind an operation to several resources, creating a set of *operation instances*, and thus introducing redundancy. Note that the third requirement ensures only at least one permutation of operation instances to work correctly and can be extended to fulfill more restricting requirements.

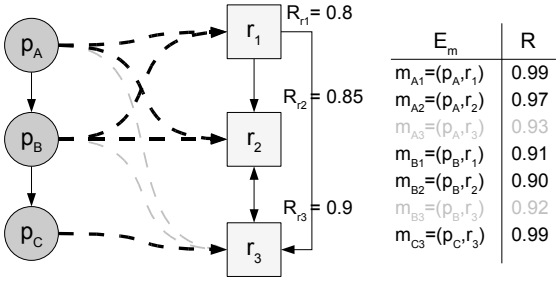

Figure 3. The black dashed mapping edges denote a feasible multi binding of a given problem graph onto a given architecture graph.

As an example, Fig. 3 shows one possible feasible binding for a given problem graph. The first requirement for a feasible binding allows operation p_A and p_B being bound onto resources r_1 and r_2 and operation p_C being bound onto r_3. Considering all resources being allocated, the second requirement is also fulfilled. The third requirement can for example be fulfilled with the set $(m_{A1} = (p_A, r_1), m_{B2} = (p_B, r_2), m_{C3} = (p_C, r_3))$. Hence, we introduced redundant instances for the operations p_A and p_B into our system, thus, increasing the reliability.

For a better visualization and for the proposed reliability analysis in section 3.2, we introduce a new type of graph called the *instance graph* $G_{\mathrm{i}} = (V_{\mathrm{i}}, E_{\mathrm{i}})$. The set of vertices V_{i} is defined as $V_{\mathrm{i}} = \beta$. An edge $e = (m, \widetilde{m}) \in E_{\mathrm{i}}$ equals to a pair of instances $m = (p, r), \widetilde{m} = (\widetilde{p}, \widetilde{r})$ with a data dependency $(p, \widetilde{p}) \in E_{\mathrm{p}}$ and the property $r = \widetilde{r} \vee (r, \widetilde{r}) \in E_{\mathrm{a}}$. This graph provides the interesting information about which permutations of operation instances lead to a correctly working system. Such a permutation can be seen as a path through the graph that includes at least one instance of every operation $p \in V_{\mathrm{p}}$. Fig. 4 shows the instance graph for the binding in Fig. 3. Next, we will show how the instance graph can be used to quantify the reliability for our system model.

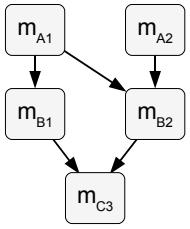

Figure 4. Instance graph for the implementation in Fig. 3

3.2. Reliability analysis

To quantify the reliability of the system, the used analysis technique has to be able to handle complex system structures. Due to resource sharing, it is not possible to simply consider combinations of series/parallel structures. But here, we want to model both, errors in the execution of an operation on a resource and the complete failure of a resource itself which is equivalent to temporary or permanents faults, respectively. To analyze the reliability in an appropriate amount of time, we introduce a technique that generates the *structure function* φ directly from the specification graph, the allocation α, and the binding β and encodes it in a *Binary Decision Diagram* (BDD) [7].

The structure function $\varphi(x)$ with $x = (a(r_1), \ldots, a(r_{|\alpha|}), a(m_1), \ldots, a(m_{|\beta|}))$ is a Boolean function. The function $a : V_{\mathrm{a}} \cup E_{\mathrm{m}} \rightarrow \{0, 1\}$ translates each activated resource and mapping into a binary variable with 1 indicating a proper operation whereas 0 indicates a resource defect or a failed computation, respectively.

$$\varphi(x) = \begin{cases} 1, & \text{system performs correct} \\ 0, & \text{system fails} \end{cases} \tag{2}$$

The function $\varphi(x)$ is constructed by Eq. (3) and (4) by using function $d : E_{\mathrm{m}} \rightarrow \{0, 1\}$. d translates operation instances into binary variables indicating whether an operation instance is in use to ensure a working system.

$$\varphi_{G_{\mathrm{s}}, \alpha, \beta}(x) = \bigexists_{m \in \beta} d(m) : \tag{3}$$

$$\bigwedge_{p \in V_\mathrm{p}} \left[\bigvee_{m=(p,r) \in \beta} d(m) \right] \ \wedge \tag{3.1}$$

$$\bigwedge_{m \in \beta} d(m) \rightarrow a(m) \ \wedge \tag{3.2}$$

$$\bigwedge_{(p,\widetilde{p}) \in E_\mathrm{p}} \ \bigwedge_{\substack{m=(p,r),\\ \widetilde{m}=(\widetilde{p},\widetilde{r}) \in \beta}} d(m) d(\widetilde{m}) \rightarrow C_{m,\widetilde{m}}(x) \tag{3.3}$$

Term (3.1) ensures that at least one[2] instance $m = (p, r)$ of each operation $p \in V_\mathrm{p}$ is working correctly and hence also the entire system. Term (3.2) ensures that the execution of an operation instance $m = (p, r)$ on resource $r \in V_\mathrm{a}$ works correctly. Term (3.3) ensures a correct communication between the operation instances with:

$$C_{m,\widetilde{m}}(x) = \begin{cases} a(r)a(\widetilde{r}), & \text{if } (m = (p,r), \widetilde{m} = (\widetilde{p},\widetilde{r})) \in E_\mathrm{i} \\ 0, & \text{else} \end{cases} \tag{4}$$

Our technique now uses the terms (3.1), (3.2) and (3.3) and generates a BDD representing the terms. After this step, the $\exists\ d(m)$ quantifier[3] is applied to the BDD for each $m \in \beta$. The resulting BDD returns true if and only if for a given $\alpha' \subseteq \alpha$ and $\beta' \subseteq \beta$ represented by the vector x, there exists at least one set of working and correct communicating operation instances, thus ensuring a correct working system.

After the BDD representing the structure function $\varphi_{G_\mathrm{s},\alpha,\beta}$ of the system G_s is generated, the BDD is used to quantify the reliability of the system by calculating the probability of the BDDs root node. For this purpose, we used a Shannon-decomposition based algorithm introduced in [12]. If the reliability of the resources and mapping edges is modeled by the failure probability which is typically the case in reliability engineering, the reliability of the root node can be used as the objective value and equals $R(t)$. Since the example in Fig.3 uses probabilities as reliability attributes, our approach can easily calculate the reliability of this implementation, which is 0.873. If the reliability of the components is given by a distribution function, e.g., *exponential distribution* or *Weibull distribution*, another value has to be chosen as the objective. We chose the *mean time to failure*, MTTF $= \int_0^\infty R(t)$, which demands a numerical integration during system synthesis. Here, the probability of the root node of the BDD is calculated for every t needed by the integration process.

4. Experimental Evaluation

In the following, we will present the experimental results of our new design methodology for the 4-tap FIR filter introduced in section 1 and a case-study of a complex automotive control system.

[2]The condition generates an 1-out-of-n redundancy and can be extended to k-out-of-n redundancies.

[3]$\exists i : f = f_{i=0} \vee f_{x=1}$

4.1. 4-tap FIR filter

As a first example, we performed a design space exploration for the 4-tap FIR filter in Fig. 1 with our methodology. The behavior and the architecture was modeled by a specification graph. In order to provide an appropriate visualization of the results, the architecture was constrained to 3 adders and 1 multiplier. Hence, if redundancy is used, we required an $\lfloor \frac{n}{2} + 1 \rfloor$-of-$n$ majority to provide fault tolerance. The objectives of the found solutions are shown in Fig. 5.

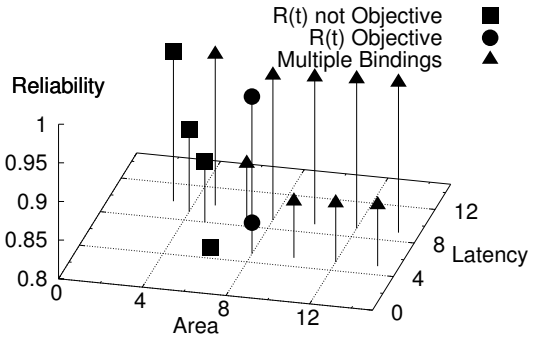

Figure 5. Optimal solutions of the 4-tap FIR filter example in the objective space.

For synthesizing the solutions pictured by rectangles, the design space methodology did not consider reliability as an objective at all. In addition to these solutions, the solutions marked by circles were found when we considered reliability as an objective during design space exploration. Integrating our multi binding technique, the optimal solutions that take advantage of redundancy are also found. A selection of especially reliable solutions is represented by triangles. Of course, our methodology treating reliability as an objective and using multiple bindings was able to find all optimal solutions including the ones found by the former explained methods.

4.2. Adaptive Light Control

In the following, we present the results of our approach applied to an industrial example from the automotive area, the so called *adaptive light control* (ALC). Consisting of 234 problem graph nodes, 1103 resources, and 1851 mapping edges, this case study allows for $\approx 2^{375}$ binding possibilities, which equals a complexity that actual design space exploration techniques can just evaluate.
Fig. 6 shows the average minimal and maximal MTTF-values over 10 explorations for the ALC. Note, that we can offer design solutions that are up to $\approx 20\%$ more reliable than the ones found by common design space exploration tools neglecting reliability. Of course, using redundancy to increase the reliability of a system creates an overhead

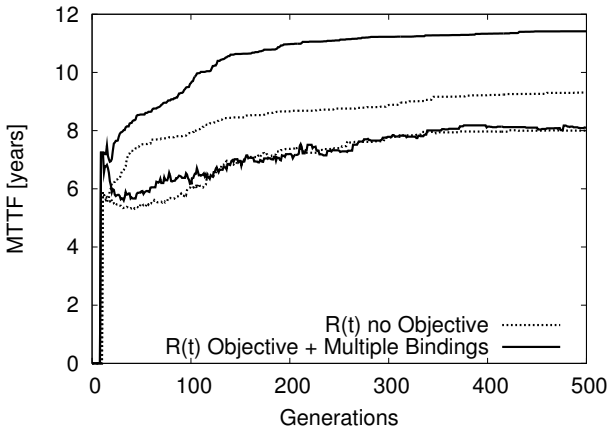

Figure 6. Adaptive Light Control: Maximal and minimal MTTF through the exploration process.

in other design objectives like area, power or cost. Hence, our approach is able to optimize this overhead in a multi-objective view. This huge example shows the ability of our reliability analysis technique to be even appropriate for the most complex examples that our exploration technique is able to handle and is therefore fully applicable in our system-level design framework.

The experiments were carried out on a Intel Pentium 4 3.20GHz machine with 1GB RAM. As an example, the average exploration time (10 exploration runs) for the ALC over 500 generations and a population size of 100 individuals was 3h37m. In our experimental results for the ALC, the exploration time was $\approx$ 10 times higher than a coarse reliability approximation by a multiplication of the reliability values. This overhead is mainly due to the complexity of constructing of the BDDs, which highly depends on the system structure and is $\mathcal{O}(2^{|\alpha|+2\cdot|\beta|})$ in worst-case.

5. Conclusions and Future Work

In this paper, we present for the first time a technique that automatically increases a system's reliability by integrating this parameter as an objective into the multi-optimization process of the design space exploration at system level. Next to an optimized allocation and the implicit usage of redundancy due to multiple binding of a process and resource reuse, our analysis technique is powerful and fast enough to handle complex systems of real world applications. Our experimental evaluations point out that we can offer a huge amount of optimal solutions with an increased reliability to the system designer automatically.

In future work, we will extend our methodology to higher abstraction layers of a system, e.g., network level. Network on chips (NoCs) for instance system provide routing features and more complex communication mechanisms between functional components.

References

[1] *Cern - Heavy Ion Collider*. http://aliceinfo.cern.ch/static/Documents/LHCC/cr5/cr5tpcfee.ppt.

[2] *Military Handbook 217FN2, Reliability Prediction of Electronic Equipment*. Department of Defense, 1995.

[3] A. Birolini. *Reliability Engineering - Theory and Practice*. Springer, 4th edition, Berlin, Heidelberg, New York, 2004.

[4] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA - A Platform and Programming Language Independent Interface for Search Algorithms. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science (LNCS)*, volume 2632, pages 494–508, Faro, Protugal, Apr. 2003.

[5] T. Blickle, J. Teich, and L. Thiele. System-level synthesis using Evolutionary Algorithms. *J. Design Automation for Embedded Systems*, 3(1):23–58, January 1998.

[6] C. Bolchini, L. Pomante, F. Salice, and D. Sciuto. Reliability Properties Assessment at System Level: A Co-Design Framework. *Journal of Electronic Testing*, 18(3):351–356, 2002.

[7] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[8] D. W. Coit and A. E. Smith. Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm. *IEEE Transactions on Reliability*, 45(1):254–260, 1996.

[9] R. Karri and A. Orailoğlu. Transformation-Based High-Level Synthesis of Fault-Tolerant ASICs. In *Proceedings of 29th Design Automation Conference (DAC'92)*, pages 662–665, USA, June 1992. ACM,IEEE.

[10] F. L. Kastensmidt, G. Neuberger, L. Carro, and R. Reis. Designing and Testing Fault-Tolerant Techniques for SRAM-based FPGAs. In *Proceedings of Computing Frontiers 2004*, Ischia, Italy, Apr. 2004. ACM.

[11] V. Kianzad and S. S. Bhattacharyya. CHARMED: A Multi-Objective Co-Synthesis Framework for Multi-Mode Embedded Systems. In *Proceedings of the 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04)*, pages 28–40, Galveston, U.S.A., Sept. 2004.

[12] A. Rauzy. New Algorithms for Fault Tree Analysis. *Reliability Eng. and System Safety*, 40:202–211, 1993.

[13] T. Schlichter, M. Lukasiewycz, C. Haubelt, and J. Teich. Improving System Level Design Space Exploration by Incorporating SAT-Solvers into Multi-Objective Evolutionary Algorithms. In *Proceedings of Annual Symposium on VLSI*, pages 309–314, Karlsruhe, Germany, Mar. 2006. IEEE Computer Society.

[14] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. Design Space Exploration of Network Processor Architectures. *Network Processor Design: Issues and Practices*, 1:55–89, Oct. 2002.

[15] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, and Y. Xie. Reliability-Centric High-Level Synthesis. In *Proceedings of Design Automation and Test in Europe (DATE'05)*, pages 1258–1263, Munich, Germany, Mar. 2005. IEEE.

[16] Y. Xie, L. Li, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Reliability-Aware Cosynthesis for Embedded Systems. In *Proceedings of ASAP'04*, pages 41–50, Sept. 2004.