Towards High-Reliability Systems Design using Agile Hardware Development Flow

J. Chen[1]*, L. Lu[1], M. Ulbricht[1], and M. Krstic[1,2]

[1]IHP– Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany
[2]University of Potsdam, Potsdam, Germany

Abstract

As transistor scaling continues to reach the deep nanometer range, modern systems face increasing challenges in ensuring reliability, particularly in safety- and mission-critical applications. Concurrently, the agile hardware development methods are gaining traction over the conventional waterfall approach in reducing hardware development costs. Therefore, investigating the intersection of highly reliable system design and agile hardware development processes is crucial. In this paper, we propose an iterative agile hardening strategy that integrates fault injection, reliability analysis, and hardening method selection. The proposed approach enables quick deployment of appropriate hardening methods and avoids over- or under-protection of the target system. Our goal is to achieve fine-grained high-reliability hardware development at a high abstraction level while balancing the trade-off between reliability, time-to-market, performance, and other important factors.

Introduction

Hardware faults are a significant concern for integrated circuits used in safety-critical and mission-critical applications, such as autonomy, aviation, and space. The demand for computing power in these applications has led to the adoption of modern processors built on advanced process technologies. However, these advanced nodes are also susceptible to faults caused by various sources, including radiation particles, voltage variations, and aging. Failure to mitigate these faults can cause system performance decline, data corruption, and even system failures, particularly in harsh conditions. Therefore, providing adequate fault mitigation methods is critical for ensuring high-reliability protection for these systems.

The field of fault mitigation has various techniques such as static or dynamic redundancy approaches in hardware, time, software, and/or information. However, applying a single hardened approach may result in either under or over-protection, along with additional overheads. Thus, using fine-grained mitigation techniques like selective hardening can be an advanced approach for designing highly reliable systems. However, implementing fine-grained hardening is usually complicated, leading to long development cycles and high costs. Furthermore, to the best of our knowledge, no designs have utilized the fine-grained hardening method in the prevalent agile hardware design flow to develop highly reliable systems.

Agile hardware development methods have been proposed to reduce the high costs associated with conventional hardware design, such as tools, labor, intellectual property, and time. These methods leverage high-level abstractions, including scala-based languages such as Chisel and SpinalHDL to decrease the cost of chip design. The benefit of using these abstraction languages is that they allow for the syntax and power of higher-level programming while still providing precise control over the generated circuits. Agile development methods allow small teams to quickly execute innovative ideas. However, the application of agile development methods to the development of high-reliability hardware has been limited.

This paper introduces a methodology for designing high-reliability systems that uses an agile hardware design flow and incorporates the fine-grained hardening approach at a higher level of abstraction. The approach includes three main steps: fault injection, model/system reliability analysis, and hardening method selection. The methodology leverages high-level abstractions tailored to the specific application requirements to enable faster iterative design cycles. This approach allows for a trade-off between system reliability, time-to-market, performance, etc.

Methodologies

The proposed methodology for designing a highly reliable system with an agile flow is depicted in Fig. 1, which consists of three iterative steps. The fault injection module rapidly simulates both transient and permanent faults at a high abstraction level for the target component. The module collects detailed fault injection data that is analyzed in the reliability analysis module to evaluate system reliability and identify the most vulnerable submodules. This analysis includes a comparison between the reliability requirements and the fault injection results. The hardening method selection module uses the results from the previous analysis to determine appropriate fine-grained hardening approaches to add or remove. The selected hardening method is then automatically integrated into the original design, resulting in a new design at the abstraction level. This new design undergoes a new iteration round to ensure that the system is neither under- nor over-protected.

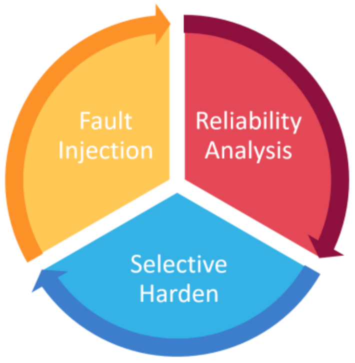

Fig. 1. Agile flow for high reliability system design.

Fault Injection

Fault injection is a prevalent technique for analyzing system behavior under faults, and it can be performed at three distinct stages: in a simulated environment, by emulating hardware in field-programmable gate arrays, or after production by exposing the fabricated die to radiation. In general, early fault injection analysis provides greater flexibility and cost-effective design decisions. Moreover, for the purpose of agile development, faults should be directly injected into the lower-level hardware description, such as Chisel, to quickly identify vulnerable components and analyze the impact of added hardening logic to mitigate failures. One existing method is the Eris [1] platform, which can analyze any RTL design that can be lowered to FIRRTL (e.g., Chisel and Verilog) and convert it to a C model. This framework offers fault injection, fault tracking, and control flow deviation detection for RTL designs using a quick C/C++ RTL simulation framework. It can identify vulnerable hardware components by randomly or selectively injecting faults and tracking their propagation through the design. Another approach is to insert transient and permanent fault models directly into the target component at the abstraction layer and analyze the component's performance in the presence of faults using a cycle-accurate simulator.

Reliability Analysis

It is the central management unit that determines the suitability of existing hardening methods and determines the vulnerable parts of the system based mainly on design requirements and fault injection results. Most current methods for estimating reliability use an architecture vulnerability factor (AVF) to identify vulnerable registers in the design by estimating the bits that may lead to incorrect execution [2]. However, this approach often overestimates the number of vulnerable registers, leading to over-protection of the target system. Therefore, it is necessary to track fault propagation during program execution and identify the registers/submodules that significantly contribute to vulnerability, which can be achieved in a higher abstraction level. After completing the analysis, the overall vulnerability indicators for the entire system and its sub-modules can be obtained and compared with the design requirements. By comparing the overall vulnerability indicators obtained from the analysis with the design requirements, it is possible to identify the sub-modules that require the most optimization, thereby minimizing the potential negative impacts of increased reliability on system performance, power consumption, and other factors.

Selective Harden

Selective hardening improves system reliability by targeting the most vulnerable components such as registers, modules, and cores, instead of hardening the entire system. Cross-layer fault tolerance [3] techniques can also be used to apply suitable hardening approaches at each system layer. For example, TMR-flip flops can replace vulnerable registers at the circuit layer, parity bits can be added to memory at the logic layer, module-level redundancy can be implemented at the architecture layer, and dynamic scheduling can be utilized at the software layer. While different hardening methods may negatively impact system parameters, selective harden with cross-layer fault-tolerant systems can harness the resources and information available at each layer to achieve improved performance, reliability, cost-effectiveness, and energy efficiency. However, determining the optimal hardening strategy may require multiple iterations and evaluation of different models.

Discussion

This paper introduces an approach to designing high-reliability systems using an agile hardware design flow. The proposed methodology incorporates the fine-grained hardening approach at a higher level of abstraction, leveraging fault injection, model/system reliability analysis, and hardening method selection. By balancing factors such as system reliability, time-to-market, and performance, designers can make informed trade-offs during the design process. This approach streamlines the design of high-reliability systems while ensuring sufficient protection.

References

[1] S. Nema, et al., "Eris: Fault Injection and Tracking Framework for Reliability Analysis of Open-Source Hardware," IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Singapore, 2022, doi: 10.1109/ISPASS55109.2022.00027.

[2] M. Maniatakos, C. Tirumurti, A. Jas, and Y. Makris, “Avf analysis acceleration via hierarchical fault pruning,” in 2011 Sixteenth IEEE European Test Symposium, 2011.

[3] E. Cheng, et al., "CLEAR: Cross-layer exploration for architecting resilience: Combining hardware and software techniques to tolerate soft errors in processor cores," 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 2016.