Evaluating Impact of Soft errors in an Embedded System

Vijay Sheshadri¹, Prasanth V²,³, Rubin Parekhji², Vishwani D. Agrawal¹, Virendra Singh³

¹Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849, USA
{vijay, agrawvd}@auburn.edu

²Texas Instruments Inc., Bangalore, India
{prasanth.v, parekhji}@ti.com

³Computer Design and Test Lab, Indian Institute of Science, Bangalore, India
viren@serc.iisc.ernet.in

Abstract

Soft errors are one of the biggest reliability challenges for present day electronic devices and their contribution to overall device failure rate is increasing with technology scaling. They have the potential to cause failures in embedded systems during run-time and are a major concern in safety critical applications, such as pacemakers. To address this issue, selective hardening techniques have been proposed in the past, aiming at providing cost-effective soft error resilience. This paper proposes that the cost-effectiveness of such a soft error mitigation technique can be further improved by taking advantage of the repeated nature of firmware execution in an embedded system and the firmware’s inherent ability to mask soft errors. This approach is applied on a module by performing fault injection analysis on the circuit during application execution. Experimental results indicate that, for the given application, bit flips in approx. 12% of the sequential logic resulted in system failures and need to be hardened against soft errors.

Keywords: Embedded systems, soft errors, application derating, fault injection

1. Introduction

Embedded systems are used for a variety of purposes in various domains today. They comprise of applications which run in the form of firmware executing on embedded hardware inside the system. These systems are often real-time in nature (e.g. electronic control circuits (ECUs) in automobiles) and have distinct reliability requirements, such as (i) Resilience to manufacturing and life-time defects. (ii) Robustness to transient errors (or soft errors) during operation. For instance, due to the strict quality requirements of automotive devices, several standards such as AEC-Q100[1], IEC61508[2] and ISO26262[3] have come up which guides the requirement in automobile electronics to detect, correct or tolerate errors before they cause a failure leading to a potential hazard. Manufacturing and life-time defects must be taken care of through use of Design For Testability (DFT) techniques and application of a large number of defect screening and life-time stress tests at manufacturing time whereas, soft errors must be taken care through design for transient error tolerance techniques, providing real-time error mitigation/correction during normal operation of the system. Some of the most common approaches to address the transient fault-tolerance involve the use of Error Correcting Codes (ECC) for memories and providing robustness in flip-flops, either in the form of new hardened flip-flop structures, such as DICE cells [4], or encoding of flip-flop groups into codewords [5]. However, since these mechanisms are based on either spatial redundancy or temporal redundancy, they incur penalties in terms of area or speed respectively. Several selective hardening techniques have been developed, at device, circuit and architecture-level, to provide a cost-effective solution to the problem of soft errors. Some of the standard selective hardening techniques include selective gate-sizing [6], selective inclusion of shadow flip-flops [5] etc.

In this work, the selection for hardening is based on: (i) soft error masking by the firmware executed on the embedded hardware. (Applications often, can intrinsically mask or recover from transient faults. This is known as application derating[7]) and (ii) the inbuilt ability of an embedded system to “tolerate” errors, owing to the qualitative interpretation of its output. (Termed as soft-computation, this concept allows reduced accuracy constraints on the results[8]). Using these concepts, we evaluate the critical components for the particular module under consideration. This evaluation is carried out by using a fault injection framework to assess the impact of random faults (similar to those created due to soft errors) during the execution of an application on this circuit in an embedded context. Components, in which the injected fault resulted in an error that results in the output lying outside the permissible tolerance band, are identified. These alone now must be hardened to mitigate soft error impact. Results show that, for the application considered, faults in about 12% of the sequential logic were found to cause errors that were beyond the defined tolerance limits, whereas failures due to faults injected in combinational logic were negligible. Although these results are application specific, they demonstrate how the inherent redundancy in repeated firmware execution can be traded off with the required hardware redundancy for an optimal implementation.

The remainder of this paper is organized as follows. Section 2 provides the background on soft errors. Section 3 details the proposed methodology for finding the critical components for a particular module and adds details of
simulations. Section 4 discusses the results for the fault injection experiment and section 5 concludes the paper.

2. Background on Soft errors

Charged particles in the atmosphere, when incident on integrated circuits can induce charge generation in the transistors and cause erroneous bit-flips upon sufficient charge collection [9]. These erroneous bit-flips are termed as Single Event Upsets (SEU) and the errors caused by them are termed as soft errors. The occurrence of soft errors is dependent on various factors such as the time instance at which the particle is incident, supply voltage, nodal capacitance etc. Scaling of transistor sizes and supply voltages in advanced technologies, although has resulted in faster and smaller ICs; it also has increased the risk of failures due to soft errors, hence necessitating design of soft error resilient circuits for improved reliability of systems [10].

![Figure 1: Possible outcomes of a random bit-flip in a system [11]](image)

A soft error in a system does not always translate to an erroneous output. The various possible outcomes of a soft error in a system are shown in fig. 1 [11]. (a) Benign faults - An erroneous bit-flip occurring in a part of the circuit that is not used by the application does not reflect at the output. (b) Detected Unrecoverable Error (DUE) – The soft error protection mechanism in the system can only detect the fault but is unable to correct it. (c) Silent Data Corruption (SDC) – The soft error in the system propagates to the output undetected and uncorrected. Soft error rates are expressed in FIT (Failure in Time). One FIT equals one erroneous bit-flip for every billion hours. The equation to calculate the FIT for a system is given as:

\[
FIT = \sum_{i=0}^{N} (\text{intrinsic error rate}, \times \text{Vulnerability factor})
\]

where ‘N’ is the number of components in the system. The vulnerability factor is a product of two terms, namely, the Timing Vulnerability Factor (TVF) which is a measure of the component’s vulnerability during each clock cycle, and the Architectural Vulnerability Factor (AVF) which is a measure of the probability of a soft error resulting in DUE/SDC. The AVF assessment of a system can be done by three ways, viz. analytical model, performance model and statistical fault injection (SFI) [11]. In this work, we will be adopting the SFI strategy for AVF estimation since it is the most widely used and accepted technique. The SFI method involves inducing a random bit flip in the hardware during run-time, typically at the RTL or gate level, to emulate a soft error and running the simulation till the end or a pre-determined point. An error is determined by comparing either the architectural state or the output of the circuit with that of an error-free circuit [12].

3. Fault Injection Simulations

The particular module selected for the analysis is duplicated and fault is injected in one instance. The outputs of both the instances are compared to check for the occurrence of error. The soft error analysis is done in three phases: pre-analysis, fault injection and error evaluation. During the pre-analysis phase, the unit under test is probed to determine the set of hardware components that will be utilized by the given application. This is done by forcing the design elements to their opposite state for the entire length of simulations. Components, in which the above experiment caused errors, are selected for the next phase, i.e., random fault
injection. On the other hand, components in which the bit-flips did not lead to errors are considered redundant/non-critical to the given application and are ignored for the rest of the experiment.

The circuit components thus obtained, are subjected to fault injection at random time intervals in the next phase. In the error evaluation phase, the output data obtained from the fault injection phase is compared with the golden value and based on the amount of deviation, the hardware units (flip-flops/logic gates) are categorized into units with no error (or zero soft error impact), units with acceptable errors (error occurred but within the tolerance limit of the particular application) and units with unacceptable errors (eligible for hardening).

The fault injection framework used in our soft error analysis is similar to the one discussed in [12, 13]. The testbench invokes the reference block and the unit under test. The fault injection module is created with the help of C & VPI (Verilog Procedural Interface) functions. Fault injection simulations for sequential logic and combinational logic are performed separately. A fault is injected, in a sequential design component, by inverting the value the node output at a random time interval. A transient fault in the combinational logic is emulated by depositing a 1ns wide pulse at the output of a simple/complex logic gate.

Since embedded systems have real-world applications, they often have relaxed precision requirements with respect to the output. They will be able to tolerate certain amount of error. Since the embedded application executes in an infinite loop, an erroneous behavior in a single iteration can ignored if the system returns to normalcy. Such errors that occur within the acceptable limits are termed as acceptable errors.

4. Results & analysis

The flip-flops are first categorized into two depending on the toggle activity as shown in Figure 3. Results for the sequential logic are divided into two categories based on the toggle activity of the flip-flop. For the low-activity flip-flop, the toggle count is between 0&2, whereas the toggle count for high-activity flip-flop is much greater than two. Low activity flip-flops can be control or configuration registers whose values may be altered once or twice or not at all altered, during the application. These can be very vulnerable to soft errors since an erroneous bit-flip may remain unchanged during the entire program. If the flip-flop is a control register, an erroneous bit-flip could disrupt the entire functionality of the system. In the case of high-activity flip-flop, however, since the flip-flop is written very often, an erroneous bit flip has a higher probability of getting overwritten.

Among the low-activity flip-flops, 83% were found to be having no impact on the output. Similarly, 58% of the high-activity flip-flops remained unused by the application. This result is summarized in figure 4.

The number of faults injected is proportional to the number of flip-flops in the group; since high activity flip-flops are less in number, the number of faults injected in them is less. Among the random faults injected in the combinational logic, only 0.4% resulted in errors; most of these errors are within the acceptable limits.

Architecture vulnerability Factor (AVF) is computed as the product of the probability that a fault injected in the component results in an error ($P_e$) and the percentage of that component with respect to total number of components. For e.g.; if a half of faults injected in a latch

![Figure 3: The categorization of FFs based on toggle activity. The low-activity FFs were found to be more in number](image-url)

![Figure 4: Outcome of fault injection experiments in sequential logic](image-url)
resulted in an error, then \( P_e = 0.5 \). Also, if 20\% of the system consisted of latches, then AVF of the latch = 0.5 * 0.2 = 0.1. Low activity flip-flops that are being utilized by the given application not only have a higher \( P_e \) but are also more in number than compared to the high-activity flip-flop and hence have a higher AVF. For the application considered in this work, the low activity flip-flops contribute up to almost 96\% of the errors in the sequential circuit; the contribution of high-activity flip-flops to the overall errors is just over 4\%. If only the low activity flip-flops were to be radiation-hardened using dual-modular redundancy, assuming that after hardening the flip-flop there will not be any soft error beyond the acceptable limit, the soft errors in the sequential logic would reduce by approx. 20x. Also, in the module under consideration, the low activity flip-flop account for 95\% of the total area of sequential circuit and approx. 12\% of the low activity flip-flop area needs to be duplicated for achieving robustness via dual-modular redundancy. Hence, the penalty is an increase in the total area by 11\% since the area of the hardened flip-flop has doubled due to the redundancy. Although these results are application specific, the worst-case scenario would only mean that all the hardware is identified as eligible for hardening. This would be the equivalent of radiation-hardening a generic system and hence soft error mitigation techniques that make use of logical/timing/electrical derating can be applied [14].

5. Conclusion

We have shown that, by exploiting the concepts of application derating and relaxed output correctness, the hardware overhead required to make a system robust against soft errors can be reduced, thus improving the cost-effectiveness of a selective hardening technique. Our results show that, for the application considered, only 12\% of the sequential logic was required to be hardened to improve the circuit’s robustness against soft-errors.

6. References