Timing Issues of Transient Faults in Concurrent Error Detection Schemes

Rodrigo Possamai Bastos, Giorgio Di Natale, Marie-Lise Flottes, Bruno Rouzeyre

Timing Issues of Transient Faults in Concurrent Error Detection Schemes

R. P. Bastos, G. Di Natale, M. L. Flottes, B. Rouzeyre
LIRMM (Université Montpellier II / CNRS UMR 5506)
Montpellier, France
{possamaiba, dinatale, flottes, rouzeyre}@lirmm.fr

***Abstract* – This work reveals additional timing difficulties by which concurrent error detection (CED) schemes can experience to deal efficiently with transients. It shows previously-unknown error scenarios where short-duration single transient faults in combinational logic circuits succeed in erroneously inverting stored results but CED schemes fail in detecting even single soft errors. The paper demonstrates that typical CED code-based schemes for protecting logic circuits are not as capable as they have been claimed whether flip-flops are used to register the error signals, and so timing conditions are suggested for a more efficient use of them.**

Keywords – transient faults; soft errors; fault attacks; concurrent error dectection codes; fault tolerance; and security

I. INTRODUCTION

IC-based systems are liable to encounter transient voltage variations induced by environmental or even intentional perturbation events. These effects – so-called *transient faults **(TFs)*** – are able to produce *soft errors **(SEs)*** by wrongly inverting stored results of circuit's operations, and so they can also make failure scenarios in fault-tolerance applications. Moreover, SE-succeeded TFs can be used as a form of fault-based attack to infer secret data during the execution of encryption operations in security applications.

Related researches until the end of 20th century were focused essentially on protecting systems against TFs arisen in memory elements, which were considered the system's most vulnerable circuits. Hence, many concurrent error detection and/or correction mechanisms were thus proposed to mitigate ***direct SEs*** induced by TFs originated in memory circuits. Nevertheless, in the last decade IC-fabrication deeper-submicron technologies as well as novel classes of malicious fault injection-based attacks – e.g. differential fault analysis (DFA) – have also pushed on the use of countermeasures against ***indirect SEs*** arisen from TFs in system's combinational logic circuits.

A TF in a system works like an extra primary input of the system's circuit. Actually, it is such as a perturbation input that can be localized in any system's part and can be fed at any instant by any kind of transient shape. Most specifically, a TF is like an asynchronous input of a certain target circuit, which is normally synchronous in most typical design cases. Therefore, a "TF-created unexpected asynchronous input" can easily violate or even cover the *latching windows **(LWs)*** of *flip-flops **(FFs)*** – i.e. a minimum period (defined as LW = set-up time + hold time) for which synchronous circuits' data must be on steady state, otherwise they would not be properly sampled. LWs make circuit's internal synchronous operations very sensitive to SE-succeeded TFs.

The traditional solution to face this issue is adding information, spatial, or time redundancy to the circuit. So if for instance a circuit's original part fails, another redundant copy permits detecting or even correcting produced errors. In theory, such redundancy-based schemes cope very efficiently with scenarios of single SEs caused by *short-duration Single TFs (i.e. **STFs** that last less time than a clock period)*, and they may not operate properly under long-duration STFs, multiple TFs, or multiple SEs. However, we reveal in this paper that timing features of a short-duration STF in logic circuits can actually provoke harmful effects at the same time upon the redundancy scheme and circuit's original parts, and so the protection can fail even in detecting a *single indirect SE (**SISE**)*.

Such a SISE-succeeded-STF-timing problem comes likely from the large need in latter years for also protecting the system's logic parts. In fact, this need has led to the development of many new mitigation mechanisms (e.g. [1][2][3][4][5]) based on ideas originally proposed either to make memory elements robust or to mitigate permanent faults. However, more complex effects of STFs in logic circuits require analysis and use of additional design timing issues that often have not been taken into account in several recent protection propositions. Hence, some typical countermeasures against SISEs may indeed not be as efficient as they seem.

Let us take a scenario of a SISE due to a STF that produces a timing problem in circuits protected by *concurrent error detection **(CED)*** codes. Fig. 1 shows a typical implementation scheme [3][4][5][6] for protecting logic circuits which uses information redundancy to make a CED and a FF to register the resultant error signal (i.e. an error flag). Fig. 2 renders timing characteristics of Fig. 1's signals under an occurrence of a STF. The STF starts at instant t_S and finishes at t_F on Logic Block's output node O_{Logic}. The clock cycle that is analyzed starts at time t_0 and finishes at t_1, and registers' FFs require a

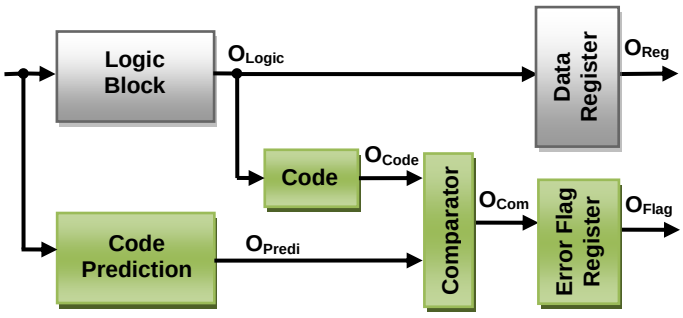

Figure 1. A state-of-the-art CED code-based scheme

set-up time $T_{Set\text{-}up}$ and a hold time T_{Hold}. Code block's delay and Comparator block's delay are respectively D_{Code} and D_{Com}.

Fig. 2's example scenario illustrates a STF on O_{Logic} covering LW. Then, a wrong data value is registered at t_1 – i.e. a SISE happens as illustrated on O_{Reg}, which stores logic value "1" instead of "0". On the other hand, Code Prediction block provides on its output O_{Predi} the correct code that should be expected from the ideal value at the output O_{Logic} under a fault-free scenario. This prediction is then compared with the code on O_{Code}, which is computed from the actual value at the output O_{Logic} under a fault scenario. If O_{Code} and O_{Predi} do not match, Comparator block's O_{Com} results in "1". Fig. 2's example therefore illustrates that the STF on O_{Logic} arises too late in the clock cycle, and so Code and Comparator blocks are not able to generate an Error Flag (i.e. "1" on O_{Com}) on time to detect SISE on O_{Reg}. In fact, as shown in the figure, the Error Flag on O_{Com} rises later than LW, and so it is not registered on Error Flag Register's O_{Flag}. Therefore, this flag on O_{Com} has not a steady condition during the clock cycle for the system deals later with. The CED scheme thus fails in detecting the SISE.

Other previously-unknown SISE scenarios and STF-timing issues that make typical CED code-based schemes inefficient are further studied in 0. The schemes' fail situations which are detailed in 0 have not yet been illustrated in the literature. Furthermore, 0 and section II of this paper discusses timing conditions for a more efficient use of CED codes.

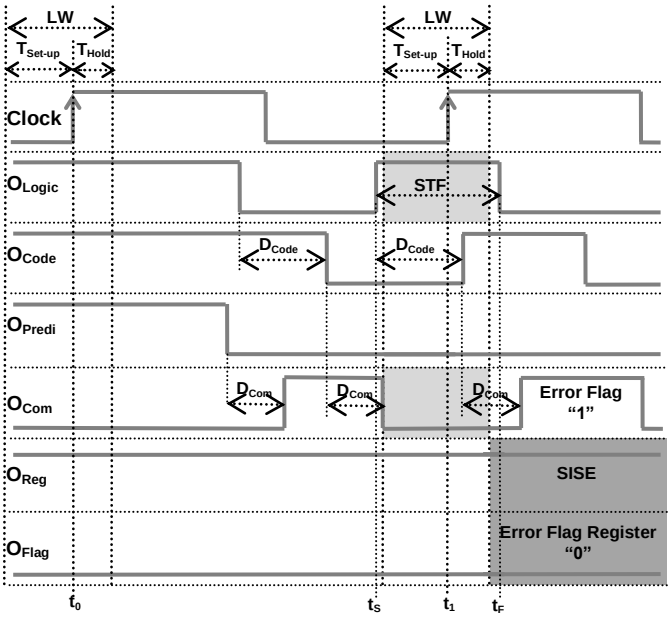

Figure 2. Timing characteristics of the CED scheme in Fig. 1

II. Conclusions and Timing Conditions for an Efficient Use of CED Codes

The vulnerability windows highlighted in this paper for Fig. 1-like scheme represent risks for operations of systems that require fault tolerance; moreover they are such as attack-prone slots which could compromise secure systems. Another more efficient scheme solution is the use of a latch with an extra clock tree instead of a FF to register the error flag, but this approach would make much more complex the IC design due to the additional clock signals. Otherwise, another alternative could be improving the efficiency of Fig. 1's scheme by minimizing the *STF-timing intervals* (***STF-TIs***) for fails – discussed in 0 – in function of fitting the delay of blocks. However, this solution would not meet better timing conditions than schemes that avoid associating their redundant parts before a timing barrier. It is not the case of Fig. 1's scheme that joins their redundant parts – i.e. Logic Block and Code Prediction block – by using the Code and Comparator blocks before the timing barrier of the Data and Error Flag Register.

In fact, this type of protection like in Fig. 1 allows a single event – i.e. a STF starting before or even during the action of registering – to wrongly affect at the same moment the redundancy scheme and circuit's original blocks. Then, the comparison mechanisms have not enough time to suitably accomplish their function. Hence, a more efficient approach is using Fig. 3's scheme which compares the results of the redundant parts after the timing barrier. The comparison mechanisms Code and Comparator blocks thus evaluate signals O_{Reg} and $O_{PrediReg}$ that have steady conditions during the clock cycle. Furthermore, otherwise Fig. 1's scheme, an eventual single direct SE, which is provoked by a STF originated in Data Register or Prediction Register, can be detected by Fig. 3's scheme. One could yet argue that any eventual STF arisen in Code or Comparator blocks could do Fig. 3's scheme not properly operating. However, such a scenario in the worst case would produce just a *false-positive error flag* (***FPEF***) 0. This Fig. 3's scheme therefore prevents the fail situations analyzed in 0, and then it is much more efficient than Fig. 1's scheme.

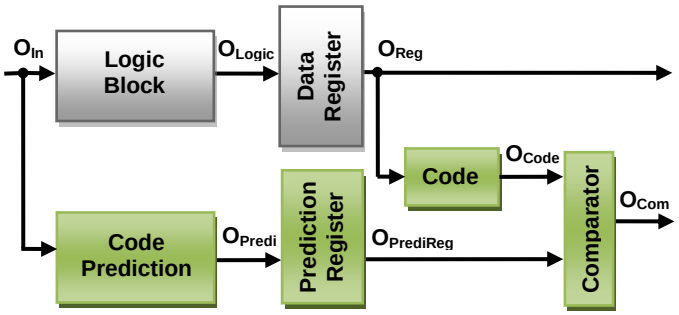

Figure 3. A more efficient state-of-the-art CED code-based scheme

References

[1] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies", in Proc. VTS, IEEE Computer Society, 1999, pp. 86-94.

[2] Y. Sasaki, K. Namba, and H. Ito, "Circuit and Latch Capable of Masking Soft Errors with Schmitt Trigger", JETTA, Kluwer Academic Publishers, v. 24, n. 1-3, pp. 11-19, 2008.

[3] M. M. Kermani, A. Reyhani-Masoleh, "Parity-Based Fault Detection Architecture of S-box for Advanced Encryption Standard", in Proc. DFT', IEEE, 2006, pp.572-580.

[4] C. Yen, B. Wu, "Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard", IEEE Transactions on Computers, v. 55, n. 6, pp. 720-731, 2006.

[5] C. Lisboa, and L. Carro, "XOR-based low cost checkers for combinational logic," in Proc. DFT, IEEE, 2008, pp. 281-289.

[6] V. Maingot, R. Leveugle, "Influence of Error Detecting or Correcting Codes on the Sensitivity to DPA of an AES S-Box", in Proc. SCS, IEEE, 2009, pp. 1-5.

[7] R. P. Bastos et al., "Timing Issues for an Efficient Use of Concurrent Error Detection Codes," in Proc. LATW, IEEE, 2011.