Timing Issues for an Efficient Use of Concurrent Error Detection Codes

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

Timing Issues for an Efficient Use of Concurrent Error Detection Codes

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 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, 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 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**)*.

Apparently such a SISE-succeeded-STF-timing problem comes 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]) based on ideas originally proposed to make memory elements robust. 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 are indeed not 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] for protecting logic circuits which uses information redundancy to make a CED. 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 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}.

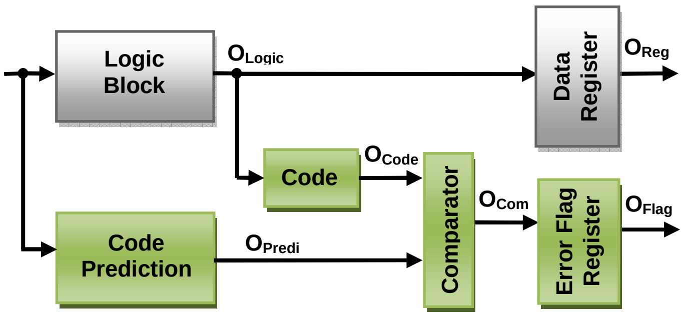

Figure 1. A state-of-the-art CED code-based scheme

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 this paper. The schemes' fail situations which are detailed in section II have not yet been illustrated in the literature. Furthermore, section III discusses timing conditions for a more efficient use of CED codes.

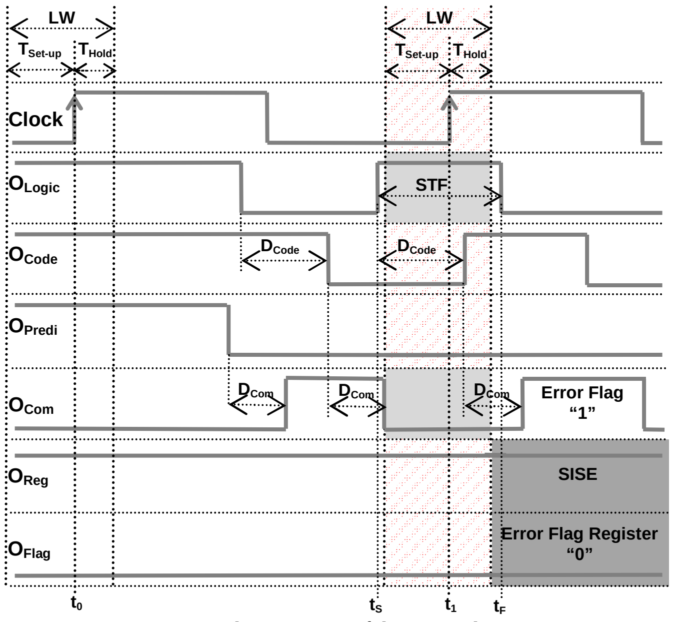

Figure 2. Timing characteristics of the CED scheme in Fig. 1

II. Fail Situations of Typical CED Schemes

A CED scheme – as any on-line testing mechanism – fails when it is not able to accomplish the results of its function on time. Essentially, any CED scheme – such as that one in Fig. 1 – provides results in accord with Table I. If Data Register is OK and Error Flag Register is "0", then the scheme is evaluated as efficient because the ideal scenario is achieved. However, if Data Register is OK and Error Flag Register is "1", it is considered inefficient in fault-tolerance applications, since an unnecessary Error Flag event is generated. On the other hand, such a *false-positive error flag* ***(FPEF)*** might be useful in security applications because this event may indicate an attempt at retrieving a secret key by means of a fault-based attack. Moreover, if Data Register contains a SE and Error Flag Register is "1", the scheme succeeds in detecting the SE. But whether Error Flag Register is "0", the scheme results in a *false-negative error flag* ***(FNEF)*** and so it fails.

Furthermore, this scheme's fail situation highly depends on the STF's timing features. In practice, a STF is classically represented as a time-varying current source characterized by a double-exponential pulse-based model, which is well discussed in [6]. However, a timing analysis of a STF that causes faulty functional behaviours in on-line testing schemes can be done through a logic abstraction-level model. In fact, this simplifies the evaluations to compare the efficiencies of the schemes by using a rectangular pulse-based model, which is detailed in [7]. This logic-level model is used in this paper to analyze fail situations of CED schemes. The circuit analysis seeks *STF-timing intervals* ***(STF-TIs)*** of *SISE-succeeded-STF prone nodes* ***(SISE-STF-PN)*** on which there are chances of the CED schemes failing in detecting SISEs arisen from STFs.

TABLE I. Evaluation of a CED Scheme by Using its Results

System	Values at Registers		Evaluation of a CED Scheme for	
Scenario	***Data***	***Error Flag***	***Fault Tolerance***	***Security***
Fault Free	OK	"0"	Efficient	Efficient
Fault	OK	"0"	Efficient	Efficient
Fault	OK	"1"	Inefficient	Efficient
Fault	SE	"1"	Efficient	Efficient
Fault	**SE**	**"0"**	Fail	Fail

A. Fail Situations of Fig. 1's Scheme

An eventual STF originated on any node of Code Prediction, Code, Comparator, or Error Flag Register blocks could only cause a FPEF. In contrast, a STF arisen in Data Register could produce a single direct SE what is classically a well-known weakness of this type of scheme as the cases of multiple TFs and multiple SEs are too. Furthermore, any scenario of STF in Logic Block that succeeds in provoking a SISE should be, in theory, detected by the scheme. However, cases of FNEF can happen as we prove below, and then the scheme in Fig.1 can fail in detecting SISEs.

Let us firstly analyze only scheme' fail cases in which a rectangular pulse, which represents a STF on a Logic Block's SISE-STF-PN, covers at least the whole duration of a flip-flop LW. Therefore, analyzing *SISE-STF-PN situations in which a Data Register's FF suffers a SISE but its LW is Not Violated* – i.e. scenarios so called as ***SISE-STF-LWNVs***. Note that the STF effects on the SISE-STF-PN O_{Logic} enclose all possible cases of SISE-succeeded-STF that could arise in Logic Block. Then, O_{Logic}'s STF-TIs on which the scheme fails in detecting SISE-STF-LWNVs are extrapolated from Fig. 2's fail situation by using different t_S and t_F as well as STF's minimum and maximum durations (TF_{Min} and TF_{Max} in (1)). $D_{Critical}$ is the delay of the circuit's critical path, D_{Logic} is the Logic Block's longest delay, and T_{Margin} is a $D_{Critical}$'s additional time margin for variations in clock operations (jitter and skew), and manufacturing and environmental variabilities [8]:

$$\begin{cases} TF_{Min} = T_{Set-up} + T_{Hold} \\ TF_{Max} = D_{Critical} + T_{Margin} \end{cases} \tag{1}$$

$$D_{Critical} = T_{Hold} + D_{Logic} + D_{Code} + D_{Com} + T_{Set-up} \tag{2}$$

Fig 2's SISE-STF-LWNV example and the conditions of scheme's fail (Table I's last row) assist us in defining O_{Logic}'s STF-TIs on which Data Register's FF suffers a SISE-STF-LWNV (*Fail Condition 1 (**FC1**)*) but Error Flag Register's FF exhibits a FNEF (*Fail Condition 2 (**FC2**)*). Then, STF-TIs of FC1 and FC2 are firstly characterized separately by finding the ranges of t_S and t_F for:

(FC1): a STF on O_{Logic} that produces SISE-STF-LWNV on O_{Reg} of the Data Register's FF – i.e. excursions of $\boldsymbol{t_S}$ (t_{S0} to t_{S2} in Fig. 3a) ***and*** $\boldsymbol{t_F}$ (t_{F2} to t_{F4} in Fig. 3b):

$$and\begin{cases} and\begin{cases} t_S > t_{S0} = (t_1 + T_{Hold} - TF_{Max}) \\ t_S < t_{S2} = (t_1 - T_{Set-up}) \end{cases} \\ and\begin{cases} t_F > t_{F2} = (t_1 + T_{Hold}) \\ t_F < t_{F4} = (t_1 - T_{Set-up} + TF_{Max}) \end{cases} \end{cases} \tag{3}$$

and (FC2): a STF on O_{Logic} which does not succeed in reaching LW of the Error Flag Register's FF, and so it generates a FNEF that is manifested by "0" on O_{Com} at least during LW to produce "0" on O_{Flag}. The FC2 excursions of t_S and t_F are derived from the ranges of t_S and t_F for a STF on O_{Logic} that does not achieve the LW of the Data Register's FF, and so it does not make a SE on O_{Reg} – i.e. excursions of $\boldsymbol{t_S}$ (t_{S0} to t_{S1} in Fig. 4a) ***and*** $\boldsymbol{t_F}$ (t_{F0} to t_{F1} in Fig. 4b) ***or*** excursions of $\boldsymbol{t_S}$ (t_{S3} to t_{S4} in Fig. 4c) ***and*** $\boldsymbol{t_F}$ (t_{F3} to t_{F4} in Fig. 4d):

$$and\begin{cases} and\begin{cases} t_S > t_{S0} = (t_1 + T_{Hold} - TF_{Max}) \\ t_S < t_{S1} = (t_1 - T_{Set-up} - TF_{Min}) \end{cases} \\ and\begin{cases} t_F > t_{F0} = (t_1 + T_{Hold} - (TF_{Max} - TF_{Min})) \\ t_F < t_{F1} = (t_1 - T_{Set-up}) \end{cases} \end{cases} \tag{4}$$

or

$$and\begin{cases} and\begin{cases} t_S > t_{S3} = (t_1 + T_{Hold}) \\ t_S < t_{S4} = (t_1 - T_{Set-up} + (TF_{Max} - TF_{Min})) \end{cases} \\ and\begin{cases} t_F > t_{F3} = (t_1 + T_{Hold} + TF_{Min}) \\ t_F < t_{F4} = (t_1 - T_{Set-up} + TF_{Max}) \end{cases} \end{cases} \tag{5}$$

Equations (4) or (5) are thus formalized only to us easily derive the STF-TIs in (6) or (7) on which a STF on O_{Logic} does the Error Flag Register's FF manifesting ***FC2***. These STF-TIs are indeed derived by just adding "– (D_{Code} + D_{Com})" to (4) and (5), since O_{Com} is delayed by "(D_{Code} + D_{Com})" from O_{Logic}:

$$and\begin{cases} and\begin{cases} t_S > t_1 + T_{Hold} - TF_{Max} - (D_{Code} + D_{Com}) \\ t_S < t_1 - T_{Set-up} - TF_{Min} - (D_{Code} + D_{Com}) \end{cases} \\ and\begin{cases} t_F > t_1 + T_{Hold} - (TF_{Max} - TF_{Min}) - (D_{Code} + D_{Com}) \\ t_F < t_1 - T_{Set-up} - (D_{Code} + D_{Com}) \end{cases} \end{cases} \tag{6}$$

or

$$and\begin{cases} and\begin{cases} t_S > t_1 + T_{Hold} - (D_{Code} + D_{Com}) \\ t_S < t_1 - T_{Set-up} + (TF_{Max} - TF_{Min}) - (D_{Code} + D_{Com}) \end{cases} \\ and\begin{cases} t_F > t_1 + T_{Hold} + TF_{Min} - (D_{Code} + D_{Com}) \\ t_F < t_1 - T_{Set-up} + TF_{Max} - (D_{Code} + D_{Com}) \end{cases} \end{cases} \tag{7}$$

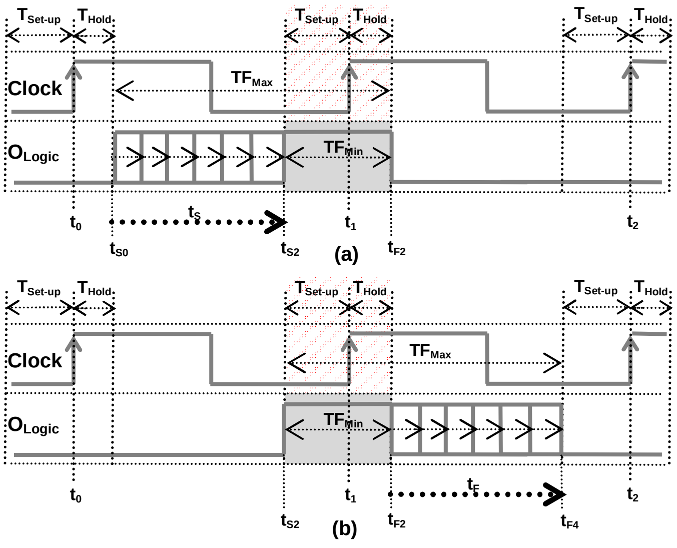

Figure 3. STF-TIs on O_{Logic} for which a Data Register's FF certainly manifest a SISE-STF-LWNV – i.e. FC1

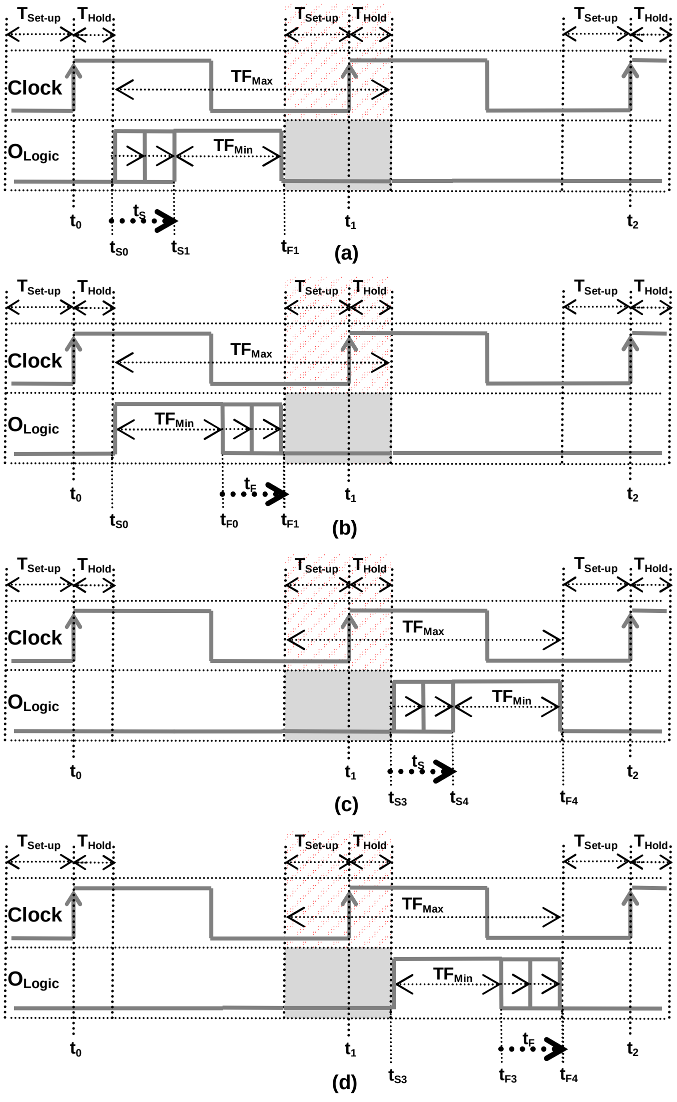

Figure 4. STF-TIs on O_{Logic} for which a Data Register's FF certainly does not manifest a SE – i.e. for deriving FC2

Finally, FC1 and FC2 have to be accomplished together to characterize STF-TIs on which the scheme fails in detecting SISE-STF-LWNVs. It is a condition required for scheme's fail (Table I's last row). Then, all FC1's STF-TIs in (3) would need to have common points with all FC2's STF-TIs in (6) or (7) to really provoke SISE-STF-LWNV scenarios.

Fig. 5 as well as Fig. 6 analyze and identify, in the most highlighted zones, such a condition of common points that generates scheme's fail situations. These zones then indicate the STF-timing conditions for SISE-STF-LWNV scenarios on the scheme, and they are characterized in (8). Fig. 1's scheme therefore certainly fails weather a STF starts (t_S) and finishes (t_F) in these STF-TIs in (8) that indeed represent only the Fig. 6's most highlighted zones, since Fig. 5 presents no common points for t_F between all (3)'s STF-TIs and (6)'s STF-TIs.

In addition to equations (6) and (7), Fig. 5 and Fig. 6 include STF-TIs in case of a STF starting or finishing within a flip-flop LW, therefore even STFs shorter than TF_{Min} are taken into account in these figures. These STF-TIs are illustrated in Fig. 5 and Fig. 6 by horizontal lines based on small squares that follow the beginning of the point-based horizontal arrows t_S and t_F, and they are defined as *SISE-STF-PN situations in which a Data Register's FF may suffer a SISE because its LW is Violated* – i.e. scenarios so identified as ***SISE-STF-LWVs***. In fact, an eventual STF in such STF-TIs would result in the metastability of the circuit, and so unknown values in registers – i.e. either an erroneous value (a SE) or the correct value.

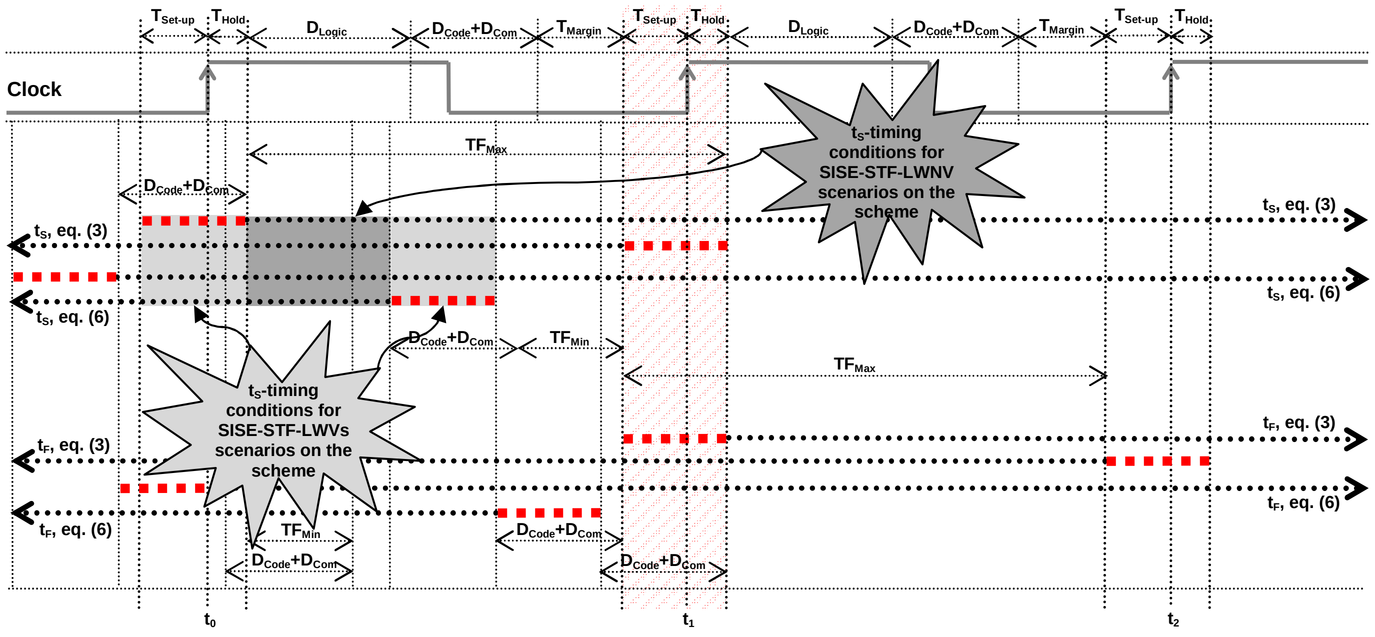
T_Set-up
T_Hold
D_Logic
D_Code+D_Com
T_Margin
Clock
TF_Max
D_Code+D_Com
t_S, eq. (3)
t_S, eq. (6)
t_S-timing conditions for SISE-STF-LWNV scenarios on the scheme
t_S-timing conditions for SISE-STF-LWVs scenarios on the scheme
TF_Min
t_F, eq. (3)
t_F, eq. (6)
t0
t1
t2

Figure 5. Equations (3) and (6): STF-TIs on which a STF on O_{Logic} does Fig. 1's scheme failing in detecting a SISE

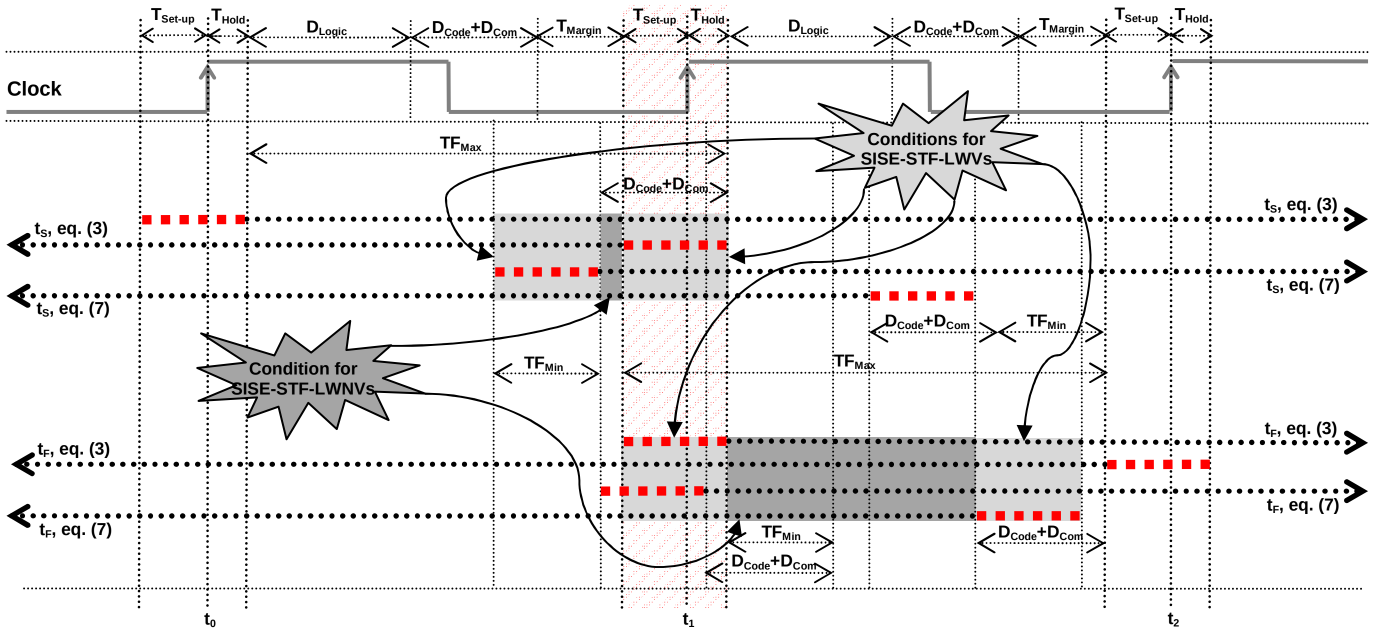

Figure 6. Equations (3) and (7): STF-TIs on which a STF on O_{Logic} does Fig. 1's scheme failing in detecting a SISE

$$and\begin{cases} and\begin{cases} t_S > t_1 + T_{Hold} - (D_{Code} + D_{Com}) \\ t_S < t_1 - T_{Set-up} \end{cases} \\ and\begin{cases} t_F > t_1 + T_{Hold} \\ t_F < t_1 - T_{Set-up} + TF_{Max} - (D_{Code} + D_{Com}) \end{cases} \end{cases} \quad (8)$$

Note that STF-TIs for fails in (8) are still longer whether Fig. 6's STF-TIs for SISE-STF-LWVs are taken into account, and so the scheme is in reality even more vulnerable to SISE-succeeded-STF scenarios. Furthermore, although Fig. 5 presents no all timing conditions that would allow scheme's fail situations, the scheme could indeed fail in case of:

$$D_{Code} + D_{Com} < T_{Set-up} + T_{Hold} \quad (9)$$

Equation (9) would allow the creation of common STF-TIs between four Fig. 5's arrows of t_F that therefore would give t_F-timing conditions for SESI-STF-LWVs scenarios on the scheme.

B. *Fail Situations of another State-of-the-Art Scheme*

Another typical CED scheme is shown in Fig. 7. It is indeed derived from [3] and Fig. 1's scheme but its $D_{Critical}$ is initially defined in (10), where D_{Extra} is the Extra Logic Block's longest delay:

$$D_{Critical} = T_{Hold} + D_{Logic} + D_{Extra} + T_{Set-up} \quad (10)$$

As before, STF-TIs on which Data Register's FF manifesting FC1 are derived from (3) by adding "– D_{Extra}":

$$and\begin{cases} and\begin{cases} t_S > t_1 + T_{Hold} - TF_{Max} - D_{Extra} \\ t_S < t_1 - T_{Set-up} - D_{Extra} \end{cases} \\ and\begin{cases} t_F > t_1 + T_{Hold} - D_{Extra} \\ t_F < t_1 - T_{Set-up} + TF_{Max} - D_{Extra} \end{cases} \end{cases} \quad (11)$$

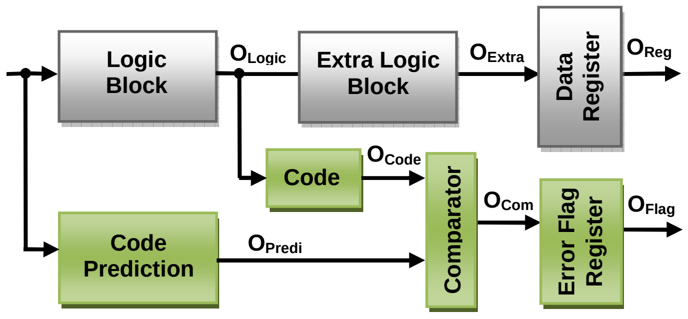

Figure 7. Another state-of-the-art CED code-based scheme

On the other hand, the same equations (6) and (7) represent STF-TIs on which the Error Flag Register's FF results in the FC2. Consequently, Fig. 8 and Fig. 9 are the counterparts of Fig. 5 and Fig. 6, and their two most highlight zones represent STF-TIs (12) and (13) on which there are chances of Fig. 7's scheme failing whether a STF happens within:

$$and\begin{cases} and\begin{cases} t_S > t_1 - TF_{Max} - T_{Set-up} - (D_{Code} + D_{Com}) \\ t_S < t_1 - T_{Set-up} - (D_{Code} + D_{Com}) \end{cases} \\ and\begin{cases} t_F > t_1 - T_{Set-up} - D_{Extra} \\ t_F < t_1 + T_{Hold} - (D_{Code} + D_{Com}) \end{cases} \end{cases} \quad (12)$$

or

$$and\begin{cases} and\begin{cases} t_S > t_1 - T_{Set-up} - (D_{Code} + D_{Com}) \\ t_S < t_1 + T_{Hold} - D_{Extra} \end{cases} \\ and\begin{cases} t_F > t_1 + T_{Hold} - (D_{Code} + D_{Com}) \\ t_F < t_1 + T_{Hold} + TF_{Max} - D_{Extra} \end{cases} \end{cases} \quad (13)$$

Furthermore, note that a SISE due to a STF arisen in Extra Logic Block is not able to be detected by the scheme whether no additional mechanisms are included to protect such a block.

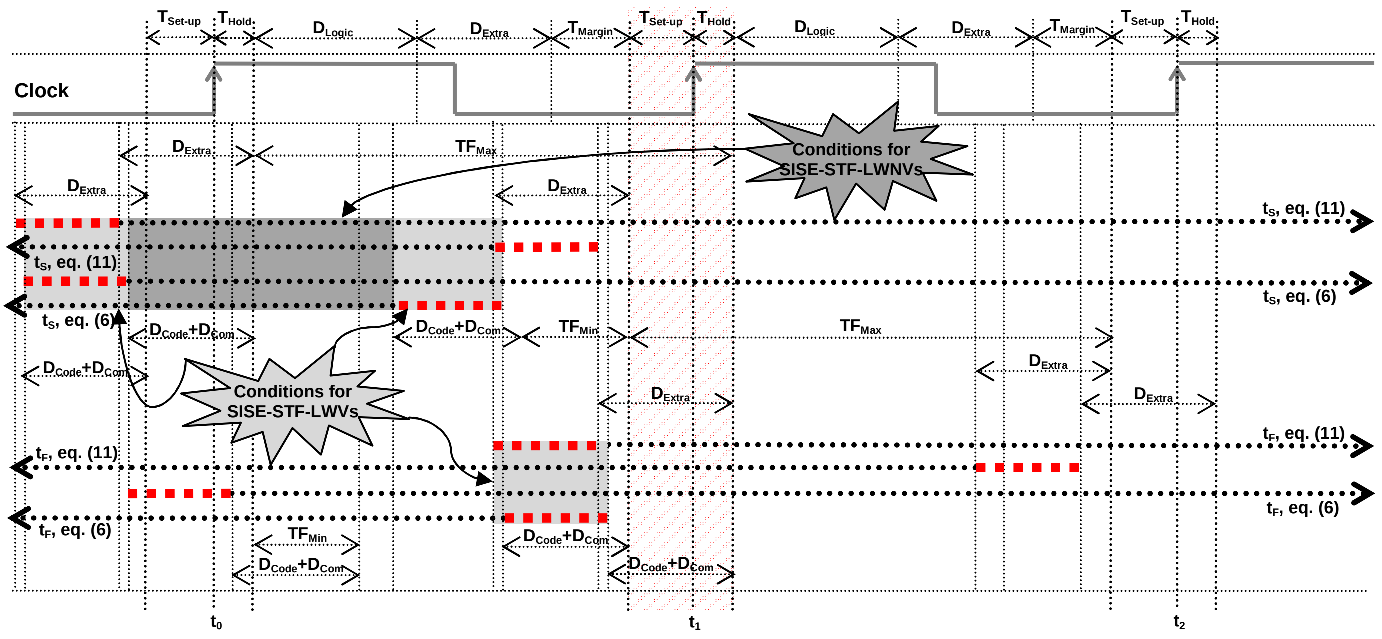

Figure 8. Equations (11) and (6): STF-TIs on which a STF on O_{Logic} does Fig. 7's scheme failing in detecting a SISE

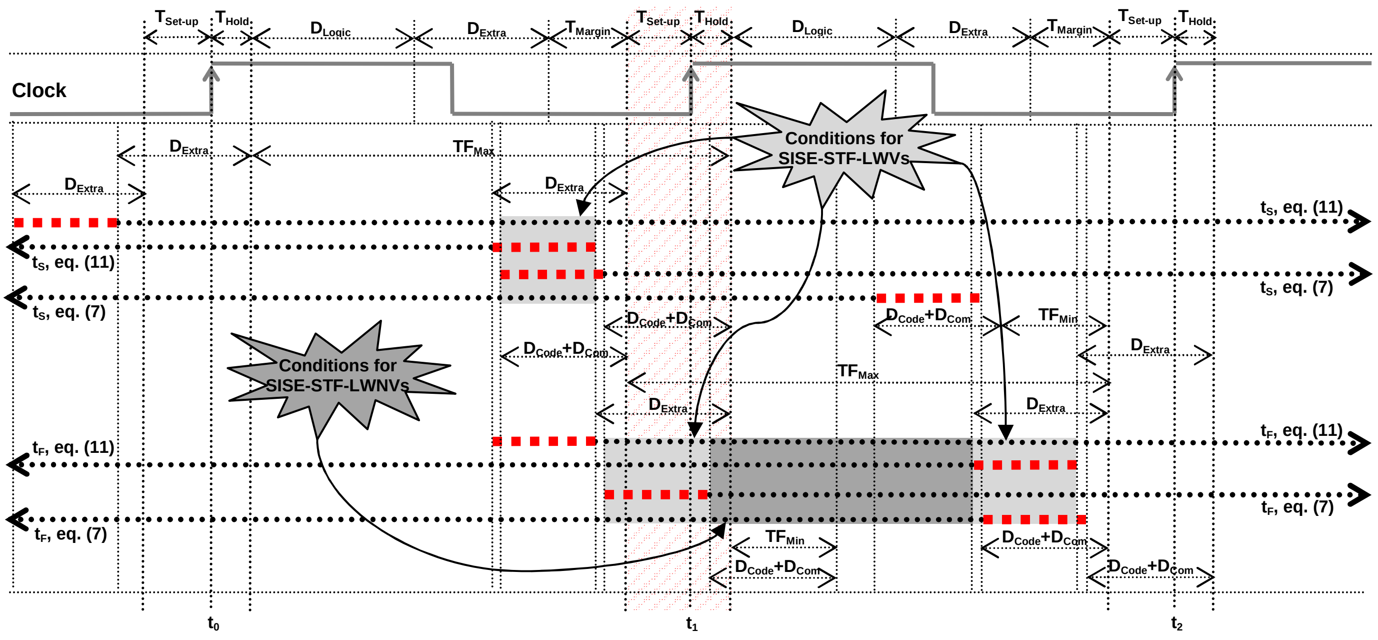

Figure 9. Equations (11) and (7): STF-TIs on which a STF on O_{Logic} does Fig. 7's scheme failing in detecting a SISE

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

The efficiency of Fig. 1 and Fig. 7's schemes may be improved by minimizing the STF-TIs for fails – discussed in section II – in function of fitting the delay of blocks. However, quite better timing conditions are met by avoiding schemes that associate their redundant parts before a timing barrier. It was the case of Fig. 1 and Fig. 7's schemes that join 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 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 solution is using Fig. 10'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 and Fig. 7's schemes, an eventual single direct SE, which is provoked by a STF originated in Data Register or Prediction Register, can be detected by Fig. 10's scheme. One could yet argue that any eventual STF arisen in Code or Comparator blocks could do Fig. 10's scheme not properly operating. However, such a scenario in the worst case would produce just a FPEF. This Fig. 10's scheme therefore prevents the fail situations analyzed in section II, and then it is much more efficient than Fig. 1 and Fig. 7's schemes. The vulnerability windows of these latter schemes represent risks for operations of systems that require fault tolerance; moreover they are such as attack-prone slots which could compromise secure systems.

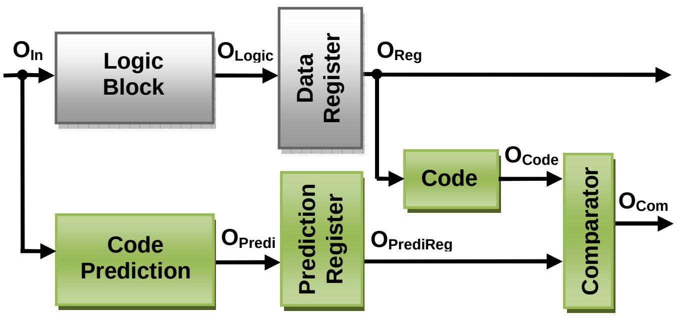

Figure 10. A more efficient state-of-the-art CED code-based scheme

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] 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.

[6] H. Cha, and J. H. Patel, "A Logic-Level Model for α-Particle Hits in CMOS Circuits", in Proc. ICCD, IEEE, 1993, pp. 538-542.

[7] D. Alexandrescu, L. Anghel, and M. Nicolaidis, "Simulating Single Event Transients in VDSM ICs for Ground Level Radiation", Journal of Electronic Testing: Theory and Applications 20, Kluwer Academic Publishers, 2004, pp. 413–421.

[8] D. Geer, "Is It Time for Clockless Chips?". IEEE Computer, v. 38, n. 3, pp. 18-21, 2005.