How to Sample Results of Concurrent Error Detection Schemes in Transient Fault Scenarios?

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

How to Sample Results of Concurrent Error Detection Schemes in Transient Fault Scenarios?

Rodrigo P. Bastos, Giorgio Di Natale, Marie-Lise Flottes, and Bruno Rouzeyre, *Members, IEEE*

***Abstract*—This work analyses and classifies strategies for sampling results of concurrent error detection (CED) schemes in transient fault scenarios. It shows that dealing with results – indicating the occurrence of transient faults in circuits – can require additional mechanisms to make the error indication useful for system's recovery procedures. The paper highlights that not all error indications are noticed by certain strategies of varied costs, and therefore their efficiencies in sampling results as well as the performance, power, and area overheads added to the CED schemes must be considered. The work then finishes presenting a qualitative comparison between existing strategies in function of design goals.**

***Index Terms*—Circuit and system radiation hardening and mitigation, concurrent error detection schemes, fault attacks, soft errors, transient faults**

I. Introduction

In the past, reliability was an issue only for safety-critical systems. Nowadays, device size reduction, power supply restriction, increased operating frequency, and high circuit density have made it a design challenge for any integrated circuit. Hence, modern systems demand higher resilience against many new issues like radiation-induced particles, aging problems, and environmental and device parameter variations. Alpha particles and cosmic neutrons, for instance, are able to generate transient voltage variations even at ground level – the so-called transient faults (TFs) that can provoke soft errors (SEs) by inverting stored results of circuit operations. Another today's related topic is the growing need for secure systems – like smartcards – where TFs can be intentionally induced as a form of fault-based attack to bypass security mechanisms and retrieve confidential information such as stored secret keys.

Related researches until early 2000's were focused essentially on protecting systems against TFs affecting memory elements, which were considered the system's most vulnerable circuits. Hence, many concurrent error detection and/or correction mechanisms were proposed to mitigate **direct SEs** – the errors induced by TFs originated in memory cells. Nevertheless, in the last decade, more sensitive deeper-submicron technologies as well as new manners of performing fault injection-based attacks – e.g. [1][2] – have also pushed for countermeasures against **indirect SEs** originated from TFs starting in system's combinational logic circuits.

Although today's ground-level systems have to be more and more TF robust, most of them certainly require more moderate levels of robustness than space applications, for instance, which are located in much more hostile environments. Hence, nowadays low-cost mitigation solutions have been optimized towards the target products and their required minimum immunity levels. Non-safety-critical systems have been thus protected with techniques of reduced redundancy and lower error detection efficiency. However, they ensure, at reasonable cost, satisfactory error coverage of the system's most recurrent operations.

Another today's trend in low-cost efficient solutions is applying protection techniques at different abstraction levels of the design [3][4][5][6]. Concurrent error detection (CED) mechanisms are designed at lower abstraction levels while system's recovery procedures (**SRP**) at higher levels. This strategy takes advantage of recovery circuits that are already present in modern microprocessors to recompute instructions in case of branch misprediction [5][6]. Then, only CED mechanisms need to be implemented, and as they are optimized at lower levels, they can monitor circuit's fault-prone nodes much closer than higher level schemes could. Hence the detection can be done early, as soon as the fault happens, preventing more critical failure scenarios such as the induction and propagation of multiple errors to other clock cycles, stages, or parts of the system. This approach, therefore, allows adding higher detection capability to the system at the expense essentially of a CED circuitry. Penalties of extra clock cycles to perform SRP occur only in faulty scenarios when the recomputation of cycles is required to correct the errors.

Classic CED solutions to face TF effects are adding spatial, information, or time redundancy to the circuit. So if for instance a circuit's original part fails, another redundant copy permits detecting produced errors by comparing both parts. The result of such a comparison reports thus an indication of error.

Manuscript received September 16, 2011.

R. P. Bastos, G. Di Natale, M. L. Flottes, B. Rouzeyre are with LIRMM (Université Montpellier II / CNRS UMR 5506), 161 rue Ada, 34095 Montpellier Cedex 5, France (phone: +33 4 67 41 86 35; fax: +33 4 67 41 85 00; e-mail: {possamaiba, dinatale, flottes, rouzeyre}@lirmm.fr).

This paper focuses essentially on CED circuitries that have ability to detect TFs started in logic blocks. We highlight that the error indications of such schemes can have behaviours as transient as the TFs which give rise to them. The error information is thus sensitive to be lost if it is not registered, and so no high-level safety reaction could be taken for protecting the system. We point out, therefore, the importance of a strategy for sampling CED's results (see Fig. 1) in order to properly ensure the activation of SRP, like clock cycle's recomputation, or other safety measure (e.g. system's restart or block of the data flow).

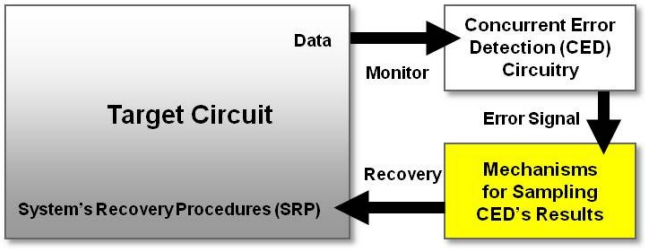

Figure 1. A target circuit protected by a CED and SRP

It is relevant to stand, to the best of our knowledge, that no work has yet analyzed comparatively the different types of strategies that make possible the communication between CED circuitries and system's reaction procedures like SRP. In section II of this paper we innovatively define two types of CED schemes. It helps us in section III and IV to analyze and classify the existing strategies for sampling CED's results. Furthermore, it allows us, in section V, estimating the system's recovery resources required by the different types of CED schemes. Finally, in section VI we notice that these strategies can impose varied overheads and different efficiencies in sampling results. We note thus that the choice of a strategy can considerably reduce inherent overheads determined by CED schemes, and it is then an important point to be also evaluated in new CED propositions.

II. Types of CED Schemes

Every CED technique employed to cope with TFs makes use of some form of redundancy such as:

1) Spatial redundancy by which the target circuit is, for instance, duplicated and an another additional module is used to identify the presence of an error;

2) Redundancy of information in which the application of error detection codes replaces the duplication of the target circuit, thus reducing surface and power consumption overheads at the cost of smaller error detection capability; and

3) Time redundancy at which circuit's operations are evaluated twice with the same data and a comparison of the results can allow identifying non-permanent faults. It attempts to reduce the amount of extra hardware at the expense of additional time.

These three approaches can be implemented at different abstraction levels of the design. Fig. 2, for instance, illustrates basic schemes of them at micro-architectural level. They essentially compare two redundant results of which at least one must be safe to permit the detection of errors. If for instance one result fails, the comparison provides an error flag, such as an error signal "O_{Com}" at logical level "1". It would indicate the occurrence of a TF in the circuit during a certain Clock Cycle that we define as **CC-Start-TF** in Fig. 3. Note that such an error flag, as a CED's result, needs to remain at a steady state enough time to be correctly dealt during a Clock Cycle Posterior to CC-Start-TF labeled as **CC-Post-TF**. It is a necessary timing condition that recovery circuits need to prepare properly, for instance, SRP until the end of CC-Post-TF. Then, the system can recompute CC-Start-TF in the following fault-free cycle that we name as **CC-SRP**.

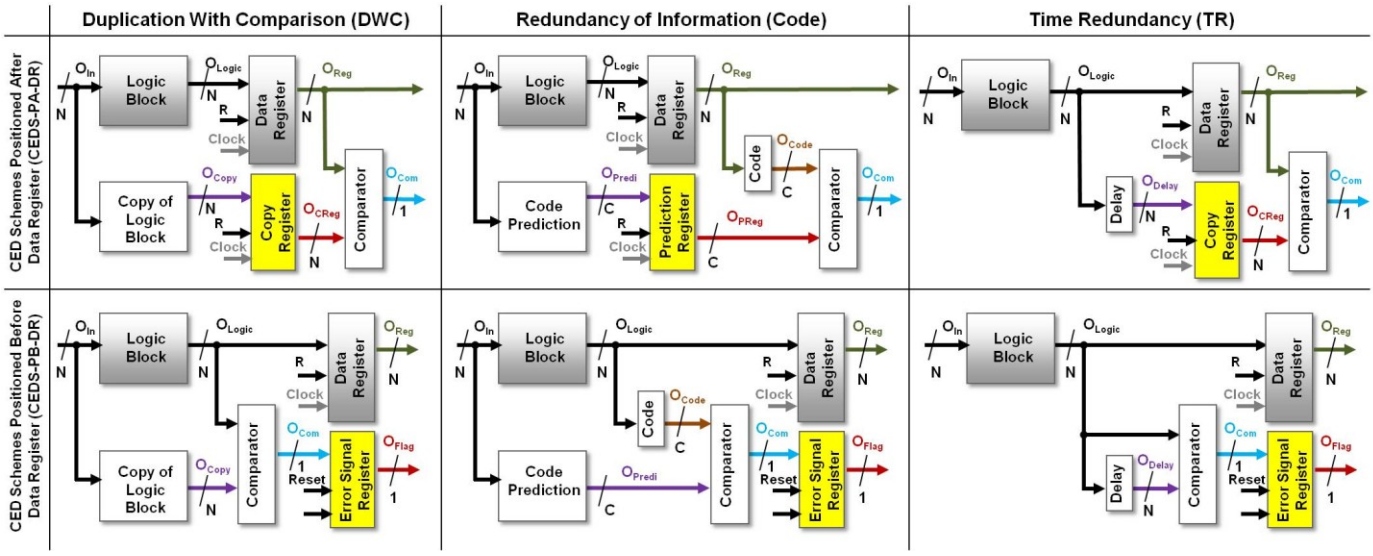

Figure 2. Basic examples of CEDS-PA-DR and CEDS-PB-DR in which O_{Com} is the error signal

In this paper we focus on CED schemes to mitigate TFs by using redundancy at micro-architectural, logical, or electrical level, and we even classify them in two types:

(1) *CED Schemes Positioned After Data Register (**CEDS-PA-DR**)*, like in [7][8][9][10][11] and Fig. 2's first row, are classic CED approaches that compare their redundant parts after the data register. Therefore, they inherently guarantee their error signals "O_{Com}" in steady conditions during all CC-Post-TF illustrated in Fig. 3. The mechanisms for sampling such CED's results are indeed parts of the CED scheme, and then no extra block is required to activate SRP or any other measure;

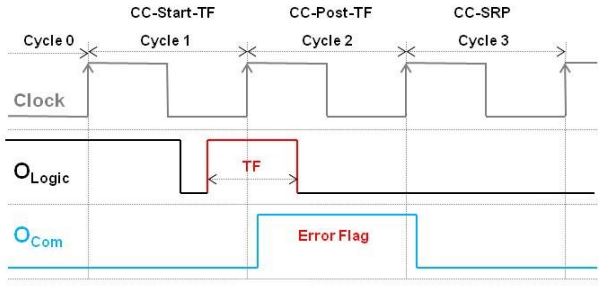

Figure 3. Signals of CEDS-PA-DR indicating an error flag due to a TF

(2) *CED Schemes Positioned Before Data Register (**CEDS-PB-DR**)*, like in [12][13][14][15][16][17] and Fig. 2's second row, have to consider the transient features of their error signals "O_{Com}". In fact, such CED's results are outputs of system's combinational logic circuits, and so they are reproduced according to the TF characteristics, see Fig. 4's example. Hence, this type of CED scheme must include another extra register, as Fig. 2 shows, dedicated only to sample their error signals "O_{Com}", and so ensuring results at steady state during a minimum time required by SRP or other measure.

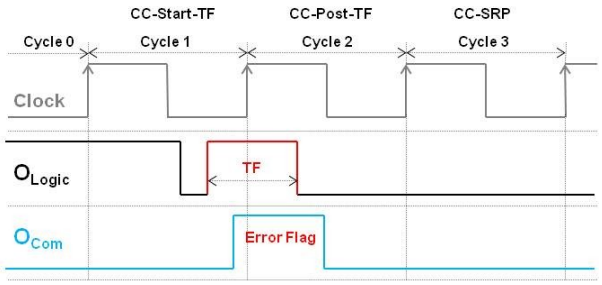

Figure 4. Signals of CEDS-PB-DR indicating an error flag due to a TF

Note that among Fig. 2's basic examples, CEDS-PB-DR are less efficient than CEDS-PA-DR since they are not able to generate an error flag when a TF arises in a register provoking a direct SE. Nevertheless, CEDS-PA-DR are more expensive in terms of area and power consumption because they require, in addition, a number N of flip-flops for Copy Register or C for Prediction Register.

III. Classic Strategy for Sampling Results of CED Schemes Positioned After Data Register

The classic strategy for sampling results of CEDS-PA-DR is taking advantage of the fact that this type of CED scheme keeps its error signals at steady state during CC-Post-TF, see Fig. 3. Then, in case of an error flag, system's reaction procedures, like SRP, have sufficient time to be configured before the end of CC-Post-TF. However, if by option or other reason the error flags are not dealt in CC-Post-TF to activate these procedures in CC-SRP, another extra register is required to just memorize CED's results, and so allowing later, in subsequent clock cycles, to proceed with such activation.

In fact, CEDS-PA-DR detect, during CC-Post-TF, direct or indirect SEs due to single TFs started in CC-Start-TF. As the assumption is the occurrence of a single fault in CC-Start-TF, there is no chance of unstable error signals at the end of CC-Post-TF since it is considered a fault-free cycle. Therefore, CEDS-PA-DR ensure stable error signals in CC-Post-TF after the comparison that is done between two redundant signals at steady state (e.g. O_{Reg} with O_{CReg} or O_{Code} with O_{PReg} in Fig. 2's first row).

This stable condition of error signals like O_{Com} in Fig. 2's first row guarantees a high efficiency in sampling error flags whether system reaction procedures are prepared in CC-Post-TF. Then, in case of single fault scenarios, classic strategy allows the sampling of any error flag issued from Comparator during CC-Post-TF. On the other hand, one could perhaps argue that any eventual single TF – arisen in Fig. 2's Code or Comparator blocks during a certain CC-Start-TF – could make CEDS-PA-DR not properly operating. However, such a scenario, in the worst case, would produce just a false-positive error flag (**FPEF**), i.e. system's reaction procedures like SRP could be activated even if Data Register was not affected by SE.

IV. Strategies for Sampling Results of CED Schemes Positioned Before Data Register

In order to deal properly with results of CEDS-PB-DR, extra mechanisms for sampling their error signals must be implemented. Otherwise the error flags are lost and more critical failure scenarios may happen, such as the induction and propagation of multiple errors to other clock cycles, stages, or parts of the system. There are three different basic strategies for sampling results that can derive other ones depending on the particularities of the CED scheme. Next subsections discuss these options initially illustrated in Fig. 5.

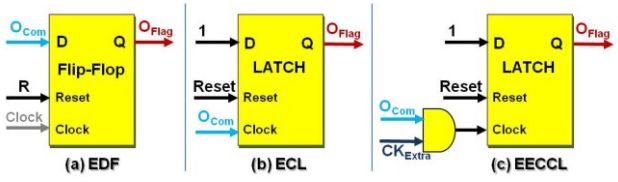

Figure 5. Strategies for sampling results (O_{Com}) of CEDS-PB-DR

A. Error Signal feeding Data Input of a Flip-flop (EDF)

The simplest design way for sampling CED's results is to use a flip-flop. The error signal of the CED scheme, such as O_{Com} in Fig. 2 and Fig. 5, feeds the data input of a flip-flop that samples it by using the circuit's global clock. No special reset is required during the operation since the error flag is normally cleaned, like in Classic strategy, by the data flow after TF vanishing.

This EDF strategy however is not able to sample error flags of some indirect-SE scenarios in which single TFs start very late in CC-Start-TF [18]. Fig. 6's example scenario illustrates this issue by assuming the code-based scheme of Fig. 2's second row. D_{Code} and D_{Com} are respectively the Code block's delay and Comparator block's delay. $T_{Set\text{-}up}$ and T_{Hold} are the set-up and hold times required by registers' flip-flops. The single TF, which starts on Logic Block's 1-bit output node O_{Logic}, covers $T_{Set\text{-}up}$ and T_{Hold}. Then, a wrong data value is registered – i.e. an indirect SE happens as illustrated on Data Register's 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. 6's example therefore shows that the TF 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. error signal "O_{Com}" at "1") on time to detect the indirect SE on O_{Reg}. In fact, as detailed in the figure, the Error Flag on O_{Com} rises later than T_{Hold}, and so it is not registered on Error Signal Register's O_{Flag}. As this flag on O_{Com} does not have a steady condition during CC-Post-TF, the recovery circuit is not able to deal with. And as this error information is not registered to be dealt in the next clock cycles, the CED scheme fails in detecting the indirect SE.

EDF strategy is therefore not so efficient for some indirect-SE scenarios, and so its efficiency in sampling CED's results is defined as moderate in relation to other strategy alternatives.

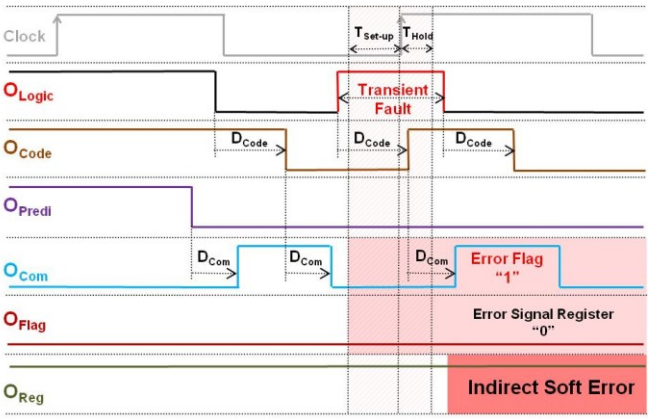

Figure 6. CEDS-PB-DR: signals of code-based scheme with EDF strategy

B. Error Signal feeding Clock Input of a Latch (ECL)

Another simple manner of sampling CED's results is to use a latch. The error signal of the CED scheme, such as O_{Com} in Fig. 2 and Fig. 5, feeds the clock input of a latch that memorizes its data input "1" (see Fig. 5) only when the error signal goes to logical level "1" (i.e. an error flag is made). The latch must be reset during the system start-up and before CC-SRP otherwise the error flag stays raised forever.

The problem of this ECL strategy is the very hard timing assumptions required for properly sampling only fault-induced error flags. In fact, all circuit paths arriving at comparator's inputs must have the same delay "$D_{paths=}$" in order to avoid the generation of glitches related to logic's delay difference (such as Fig. 7 shows for Fig. 2's code-based CEDS-PB-DR approach). Otherwise, these logic-related glitches would be considered as FPEFs, and the latch, which works such as the Error Signal Register in Fig. 2, would indicate a SE occurrence even in an error-free scenario.

This problem is even more critical because logic-related glitches are very common in every clock cycle of synchronous circuit, and today's fabrication process variability on circuit paths as well as complex wire distributions further induce these static hazard events. Therefore, CED's results could indicate systematically FPEFs, such as in every clock cycle, even in fault-free scenarios. In reality, as this ECL strategy is not able to distinguish a logic-related glitch from a fault-induced error flag, the system would be continuously alarmed with an error flag, and so it would not run appropriately its data. Moreover, even so in a remote case the hard timing assumptions of all paths with the same delays were successfully met, there would be always an imminent risk of glitches due to the process variability on such delays. This ECL strategy is thus not recommended since its efficiency in properly sampling real CED's results of CEDS-PB-DR is very low.

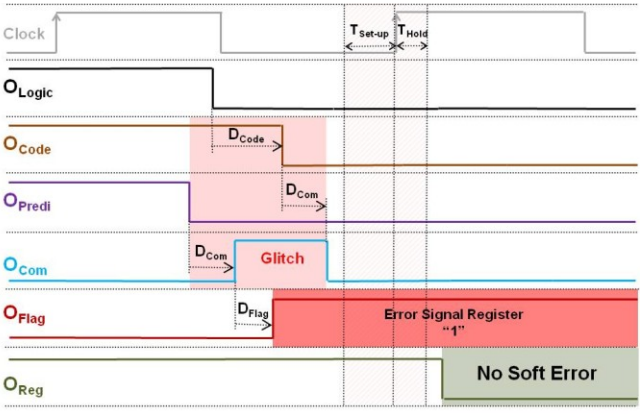

Figure 7. CEDS-PB-DR: signals of code-based scheme with ECL strategy

Nevertheless, similar strategy – without requiring timing assumptions – can provide high efficiency in sampling error flags whether particular CEDS-PB-DR like Bulk Built-In

Current Sensors (**BBICS**) [6][17] are used instead of Fig. 2's schemes or their derived ones. The error signals in BBICS-based approaches come from the activity of bulk nodes instead of data nodes, and so there is no chance of logic-related glitches being considered as FPEFs.

C. Error Signal and Extra Clock feeding Clock Input of a Latch (EECCL)

Previous ECL strategy can become highly efficient for sampling results of comparator-based CEDS-PB-DR whether an extra clock CK_{Extra} is used as Fig. 5 and Fig. 8 illustrate. The latch thus would memorize its data input "1" (Fig. 5) just if error signal "O_{Com}" and CK_{Extra} are "1". CK_{Extra} would do the latch sampling the error signal "O_{Com}" only during an interval PW_{ckEx} on which the conditions of no legal transaction events and no logic-related glitches are ensured by adding extra timing assumptions to the system design. Then, any event "1" on error signal "O_{Com}" during PW_{ckEx} is certainly a fault-induced error flag and not a legal transaction event or a logic-related glitch.

This EECCL strategy derived from [15] is clearly the most complex in terms of IC design since it requires several extra timing assumptions that are also difficult to meet.

The first timing assumptions are related to CK_{Extra}, which needs of an extra clock-tree implementation and has to be delayed by a time "D_{ckEx}" from the global clock's rising edge, such as Fig. 8 shows. D_{ckEx} is given in function of the latch's minimum pulse width "T_{MinPW}"; the set-up time "$T_{Set\text{-}up}$" required by registers' flip-flops; and the CED's delay "D_{CED}" that is equal to D_{Code} + D_{Com} for Code scheme and D_{Com} for DWC or TR scheme:

$$D_{ckEx} = D_{CED} - T_{Set-up} - T_{MinPW} \quad (1)$$

In fact, (1) is expressed by using the time "D_{CED}" between the TF's first possible edge that is able to violate $T_{Set\text{-}up}$ and its consequent error flag's falling edge. D_{CED} must be even reduced by $T_{Set\text{-}up}$ to make as reference the global clock's edge. And a margin "T_{MinPW}" must be used to ensure the sampling of such a consequent error flag from the TF's first possible edge. Moreover, a minimum slack "T_{MinPW}" is added to clock period for preventing that legal transaction events cause systematically FPEFs. Therefore, the circuit's minimum clock period "$P_{MinClock}$" is defined in function of Logic Block's longest delay "D_{Logic}", T_{MinPW}, $T_{Set\text{-}up}$, T_{Hold}, and a T_{MinCk}'s additional time margin "T_{Margin}" for variations in clock operations (jitter and skew), and manufacturing and environmental variabilities:

$$P_{MinClock} = T_{Hold} + D_{Logic} + T_{MinPW} + T_{Set-up} + T_{Margin} \quad (2)$$

Differently, note in Fig. 2 that $P_{MinClock}$ for CEDS-PA-DR and CEDS-PB-DR using EDF strategy is probably longer, since D_{CED} is normally greater than T_{MinPW}:

$$P_{MinClock} = T_{Hold} + D_{Logic} + D_{CED} + T_{Set-up} + T_{Margin} \quad (3)$$

Another option, instead of using (1) as a timing assumption for EECCL strategy, would be setting D_{ckEx} equal to zero. Then, a greater slack "D_{CED} – $T_{Set\text{-}up}$" would be required to avoid that legal events generate FPEFs. It would most likely impose worse performance penalty than (2) but better than (3):

$$P_{MinClock} = T_{Hold} + D_{Logic} + D_{CED} + T_{Margin} \quad (4)$$

Furthermore, as EECCL as ECL strategy requires timing assumptions related to the circuit paths' shortest delay "D_{Min}":

$$D_{Min} > T_{Hold} + D_{CED} + T_{MinPW} \quad (5)$$

It ensures – during an initial period "D_{Min}" in CC-Post-TF – that there are no results from new values at Logic Block's inputs which could suppress error flags. Assumption (5) is indeed the necessary minimum time to properly identify the error flag from the TF's last possible edge that is able to violate T_{Hold}. It therefore guarantees that error flags are conserved during enough time in CC-Post-TF to detect TFs which potentially would cause indirect SEs at the end of CC-Start-TF.

Finally, if (2) is used, PW_{ckEx} is defined reducing (5) by (1). Then, the range from the TF's first possible edge to the last one that is able to violate the flip-flop's latching window ($T_{Set\text{-}up}$ + T_{Hold}) is taken into account:

$$PW_{ckEx} = T_{Set-up} + T_{Hold} + 2 \cdot T_{MinPW} \quad (6)$$

On the other hand, if (4) is used, PW_{ckEx} is derived only from (5) because D_{ckEx} is equal to zero:

$$PW_{ckEx} = T_{Hold} + D_{CED} + T_{MinPW} \quad (7)$$

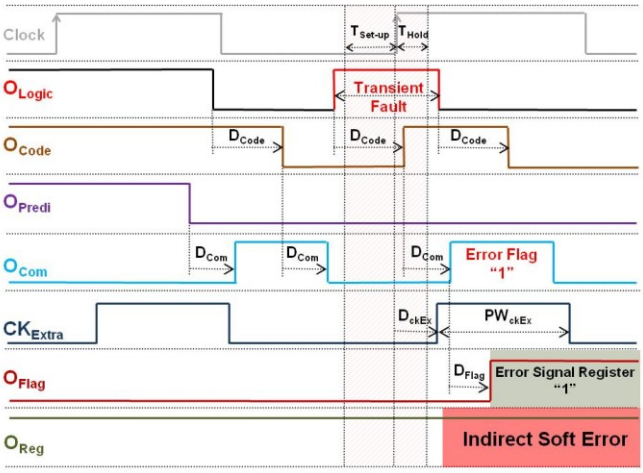

Figure 8. CEDS-PB-DR: signals of code-based scheme with EECCL strategy

V. System's Recovery Resources Required by Protection Techniques Based on CED and SRP

Protection techniques based on CED and SRP require recovery circuits to save input status of CC-Start-TF as well as to reload them, after TF vanishing, at beginning of CC-SRP. Thus, during such a recovery cycle "CC-SRP", the faulty cycle "CC-Start-TF" can be reproduced without the fault that generated the error flag.

Fig. 9 and Fig. 10 illustrate the two types of CED schemes (defined in section II) interacting with a recovery circuit to protect logic blocks and registers. Note that the communication between CED circuitries and recovery circuits are slightly different. More precisely, the type of CED scheme and the strategy for sampling its results determine which minimum recovery resources are necessary to properly protect the system.

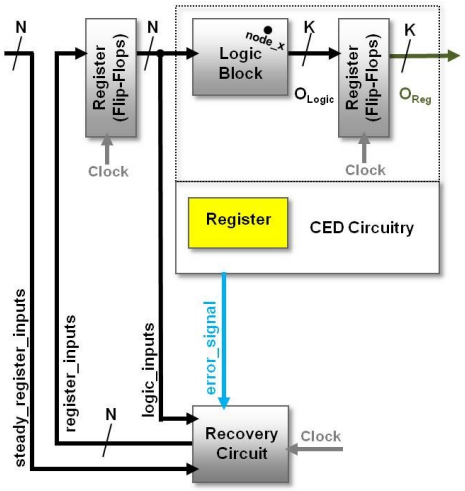

Figure 9. Logic blocks and registers protected by technique based on CEDS-PA-DR and SRP

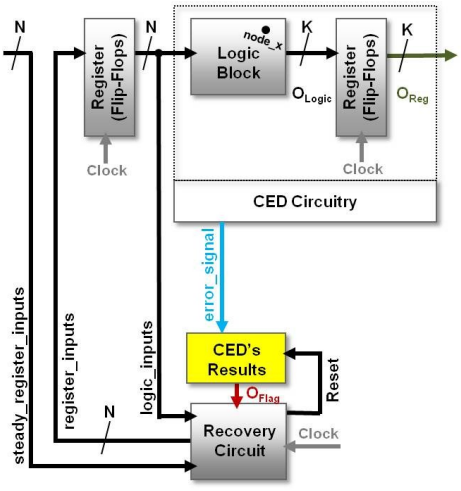

Figure 10. Logic blocks and registers protected by technique based on CEDS-PB-DR and SRP

CEDS-PA-DR require at least a recovery circuit similar to Fig. 11's illustration to efficiently activate SRP. This machine saves input status (logic_inputs) in each clock cycle by using a file with N latches. Then, such as Fig. 3's example, if CED circuitry indicates in CC-Post-TF an error flag at error_signal, the recovery circuit is able to restore in CC-SRP saved input status (saved_logic_inputs) of state executed one clock cycle ago the instant at which the error flag is set (i.e. CC-Start-TF).

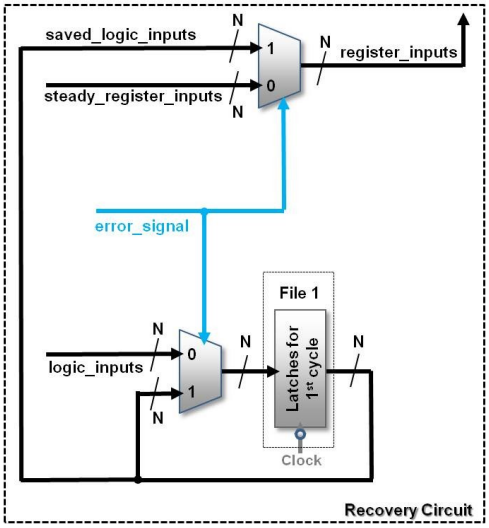

Figure 11. Recovery circuit for CEDS-PA-DR and CEDS-PB-DR with EDF strategy in which error_signal is replaced by O_{Flag}

CEDS-PB-DR using EDF strategy also demand recovery circuits like the schematic in Fig. 11, but O_{Flag} is connected to multiplexers instead of error_signal. However, CEDS-PB-DR with ECL or EECCL strategy require a more elaborate recovery architecture such as Fig. 12 shows. Otherwise, this machine saves logic_inputs of two clock cycles by using two files with N latches each one. Thereby, if CED circuitry indicates an error flag at error_signal (O_{Flag} at "1" as Fig. 13 illustrates), the recovery circuit is able to restore in CC-SRP saved input status (saved_logic_inputs) of a state executed two clock cycles ago the instant at which the error flag is identified and registered at recomputing_signal (i.e. CC-Start-TF).

CEDS-PB-DR with ECL or EECCL strategy require more recovery resources than the other approaches essentially because their error signals act as a recovery circuit's asynchronous primary input. In fact, as the error signals can have the same transient features like the natural behavior of the TFs, they can happen with any duration and at any instant as an asynchronous event. Then, another flip-flop, as illustrated in Fig. 12, is mandatory to accurately synchronize the error signals with the recovery circuit. Even so the error signals are already in steady state due to ECL or EECCL strategy, they require Fig. 12's flip-flop to prevent metastability problems. This flip-flop also ensures enough time to reset Error Signal Register as well as it deals with cases in which the response time "RT" is longer than clock's high pulse width.

CEDS-PB-DR with ECL or EECCL strategy also demand a multiplexer, as Fig. 12 shows, to reset their latches used as Error Signal Register. Moreover, they necessarily need at least two files with N latches to save input status of two clock

cycles. As Fig. 13 highlights, there are chances of TFs starting in Cycle 1 not to raise recomputing_signal in Cycle 2, and then the input status of Cycle 1 must be saved and available in Cycle 3. Furthermore, if RT defined by D_{CED} + Error Signal Register's delay "D_{Flag}" is greater than the clock period (e.g. [6]'s BBICS), more than two file are required. Therefore, the slower the RT the greater is the number of required files. The major problem is when the recovery circuit has no a minimum number of files to properly accomplish SRP.

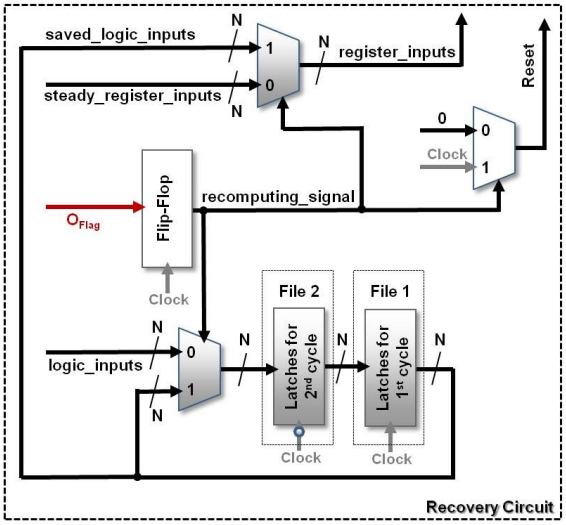

Figure 12. Recovery circuit for CEDS-PB-DR using ECL or EECCL strategy

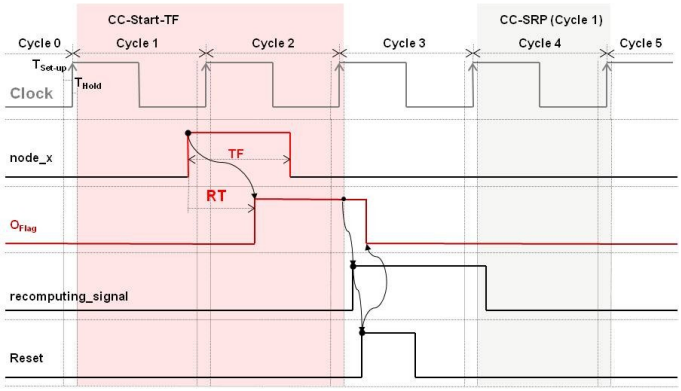

Figure 13. Signals of CEDS-PB-DR with ECL or EECCL strategy interacting with a recovery circuit

Finally, note in Fig. 11 and 12 that the area overheads related with recovery circuits are minimal whether the target circuit's architecture has already a machine to repeat operations in branch-misprediction situations. The highlighted blocks in Fig. 11 and 12 are resources that might be already present in certain modern architectures. Furthermore, performance costs through extra clock cycle executions to recompute occur only when a fault is detected, and so the circuit can operate with negligible penalty in fault-free scenarios.

VI. Analysis of Strategies for Sampling CED's Results and Major Conclusions

In this paper we have classified and compared the strategies for sampling results of CED schemes. Section III's classic strategy is highly efficient in sampling error flags originated from single TFs because CEDS-PA-DR ensure their error signals in steady condition during a fault-free cycle (CC-Post-TF). However, this condition is satisfied only by using a number N or C of flip-flops before the Comparator, as illustrated in Fig. 2's first row. On the contrary, sampling results of CEDS-PB-DR requires an extra 1-bit register to make their error signals stable, and then EDF, ECL, or EECCL strategy must be used.

EDF strategy uses only one flip-flop as Error Signal Register but it is not as efficient as the classic one since some scenarios of single TFs that cause indirect SEs can occur out of the Error Signal Register's sample window. On the other hand, ECL approach is indeed not applicable to comparator-based CEDS-PB-DR although similar strategy works very efficiently in BBICS-like CED solutions [6], which monitor bulk nodes instead of logic-related glitch prone data nodes. Otherwise EECCL strategy – at the expense of an AND gate and an extra clock tree – has a high efficiency in sampling results of comparator-based CEDS-PB-DR. Both ECL and EECCL strategies use only one latch for sampling error flags and, as explained in section IV.C, the timing assumption of D_{Min} need to be implemented by using additional logic gates. Hence, these strategies have further area costs "$C_{Dmin>}$" to meet such an assumption. ECL strategy also induces area costs "$C_{Dpaths=}$" for approaching circuit paths' delays to $D_{paths=}$, which is defined in section IV.B.

Table I summarizes the analysis done in this paper such as the area overheads and efficiencies of classic, EDF, ECL, and EECCL strategies. We can conclude that classic and EECCL strategies are the most efficient and the most expensive options, while EDF strategy is the cheapest alternative at the price of sacrificing its efficiency. Furthermore, Classic and EDF are the strategies that require less recovery resources. And EECCL makes much more complex the IC design due to the extra clock tree and its hard timing assumptions. Therefore, EDF is the strategy for implementations that aim low area with low design efforts, while the classic one is useful when the target is high efficiency with low design efforts. On the other hand, EECCL strategy is interesting in terms of area and efficiency for combinational logic circuits with several output bits, small $C_{Dmin>}$, and if recovery circuit contains 2 files.

References

[1] C.N. Chen, and S.M. Yen, "Differential Fault Analysis on AES Key Schedule and Some Countermeasures," in Proc. ACISP, vol. 2727 of LNCS, 2003, pp 118-129.

[2] P. Dusart, G. Letourneux, and O. Vivolo, "Differential Fault Analysis on A.E.S," in Proc. ACNS, vol. 2846 of LNCS, 2003, pp 293-306.

[3] C. Lisboa, M. Erigson, and L. Carro, "System level approaches for mitigation of long duration transient faults in future technologies," in Proc. ETS, IEEE, 2007, pp. 165-170.

[4] C. Albrecht et al., "Towards a Flexible Fault-Tolerant System-on-Chip," in Proc. ARC, VDE Verlag GMBH, 2009, pp. 83-90.

[5] S. Z. Shazli, M. B. Tahoori, "Transient Error Detection and Recovery in Processor Pipelines," in Proc. DFT, IEEE, 2009, pp. 304-312.

[6] C. Lisboa et al., "Using Built-in Sensors to Cope with Long Duration Transient Faults in Future Technologies," in Proc. ITC, IEEE, 2007, pp. 1-10.

[7] S. Mitra and E. McCluskey, "Which concurrent error detection scheme to choose?," in Proc. ITC, IEEE, 2000, pp. 985–994.

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

[9] Anghel, L., and M. Nicolaidis, "Cost Reduction and Evaluation of a Temporary Faults Detecting Technique," in Proc. DATE, IEEE, 2000, pp. 591-598.

[10] D. Ernst et al., "Razor: A low-power pipeline based on circuit-level timing speculation," in Proc. MICRO, IEEE/ACM, 2003, pp. 7-18.

[11] K. Bowman et al., "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," IEEE JSSC, v. 44, n. 1, pp. 49–63, Jan. 2009.

[12] S. Das et al., "RazorII: In situ error detection and correction for PVT and SER Tolerance," IEEE JSSC, vol. 44, no. 1, pp. 32–48, Jan. 2009.

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

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

[15] D. Rossi, M. Omanã, and C. Metra, "Transient fault and soft error on-die monitoring scheme," in Proc. DFT, IEEE, 2010, pp. 391–398.

[16] D. J. Palframan, N. S. Kim, M. H. Lipasti, "Time Redundant Parity for Low-Cost Transient Error Detection," in Proc. DATE, IEEE, 2011.

[17] E. H. Neto et al., "Using Bulk Built-in Current Sensors to Detect Soft Errors," IEEE Micro, vol. 26, no. 5, pp. 10–18, Sep. 2006.

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

TABLE I

	CED Scheme	Strategy for Sampling CED's Results				
		Classic	EDF	ECL	EECCL ($D_{ckEx} > 0$)	EECCL ($D_{ckEx} = 0$)
Paper's Section	II	III	IV.A	IV.B	IV.C	
Estimated Area Overhead due to the CED scheme	DWC	Copy of Logic Block + N·Flip-Flop(s) + Comparator	Copy of Logic Block + 1·Flip-Flop(s) + Comparator	Copy of Logic Block + 1·Latch +$C_{Dpaths=}$+ $C_{Dmin>}$ + Comparator	Copy of Logic Block + 1·Latch + 1·AND + Extra Clock Tree + $C_{Dmin>}$ + Comparator	
	Code	Code Prediction + C·Flip-Flop(s) + Code + Comparator	Code Prediction + 1·Flip-Flop(s) + Code + Comparator	Code Prediction + 1·Latch +$C_{Dpaths=}$+ $C_{Dmin>}$ + Code + Comparator	Code Prediction + 1·Latch + 1·AND + Extra Clock Tree + $C_{Dmin>}$ + Code + Comparator	
	TR	Delay + N·Flip-Flop(s) + Comparator	Delay + 1·Flip-Flop(s) + Comparator	Delay + 1·Latch +$C_{Dpaths=}$+ $C_{Dmin>}$ + Comparator	Delay + 1·Latch + 1·AND + Extra Clock Tree + $C_{Dmin>}$ + Comparator	
Estimated Area Overhead due to the Strategy	DWC	N·Flip-Flop(s)	1·Flip-Flop	1·Latch +$C_{Dpaths=}$+ $C_{Dmin>}$	1·Latch + 1·AND + Extra Clock Tree + $C_{Dmin>}$	
	Code	C·Flip-Flop(s)				
	TR	N·Flip-Flop(s)				
Mitigation of Direct SEs?	DWC	Yes	No			
	Code					
	TR					
	BBICS	n/a		Yes	n/a	
Mitigation of Indirect SEs?	DWC	Yes				
	Code					
	TR					
	BBICS	n/a		Yes	n/a	
Efficiency in Sampling Error Flags due to Single TFs	DWC	High	Moderate	Very Low	High	
	Code					
	TR					
	BBICS	n/a		High	n/a	
Extra Timing Assumptions?	DWC	No		$D_{paths=}$; D_{Min}	D_{Min}; D_{ckEx}; PW_{ckEx}; Extra Slack	
	Code					
	TR	D_{Delay_Block}	D_{Delay_Block}	D_{Delay_Block}; $D_{paths=}$; D_{Min}	D_{Delay_Block}; D_{Min}; D_{ckEx}; PW_{ckEx}; Extra Slack	
	BBICS	n/a		Calibrating RT	n/a	
Penalty in Performance (Fault-Free Scenarios)?	DWC	No	D_{Com}	D_{Com}	T_{MinPW}	$D_{Com} - T_{Set-up}$
	Code		$D_{Code} + D_{Com}$	$D_{Code} + D_{Com}$	T_{MinPW}	$D_{Code} + D_{Com} - T_{Set-up}$
	TR	D_{Delay_Block}	$D_{Delay\_Block} + D_{Com}$	D_{Com}	T_{MinPW}	$D_{Com} - T_{Set-up}$
	BBICS	n/a		No	n/a	
Penalty in Performance (Short-Duration TF Scenarios)?	DWC	2 Extra Clock Cycles		2 or 3 Extra Clock Cycles	2 or 3 Extra Clock Cycles	
	Code					
	TR					
	BBICS	n/a			n/a	
Recovery Resources in Short-Duration TF Scenarios	DWC	2·Multiplexers +1·File		1·Flip-Flop + 3·Multiplexers + 2·Files	1·Flip-Flop + 3·Multiplexer + 2·Files	
	Code					
	TR					
	BBICS	n/a			n/a	
Strategies in Function of Design Goals	VI	High efficiency; Low design efforts; Small recovery circuit; If DWC or Code: High performance.	Low area; Low design efforts; Small recovery circuit.	If BBICS and 2 available files: Low area; High efficiency.	If several logic outputs, small $C_{Dmin>}$, and 2 available files: Low area; High efficiency.	