Temporal Redundancy Latch-based Architecture for Soft Error Mitigation

Robert Schmidt*, Alberto García-Ortiz†, Goerschwin Fey*†
* Institute of Space Systems, German Aerospace Center, 28359 Bremen, Germany {robert.schmidt,goerschwin.fey}@dlr.de
†University of Bremen, 28359 Bremen, Germany, agarcia@item.uni-bremen.de fey@informatik.uni-bremen.de

***Abstract*—Current transients caused by energetic particle strikes are a serious threat for digital circuits in aerospace applications. Such single-event transients (SETs) can corrupt the circuit state, with possibly devastating consequences. Although it is possible to protect circuits with spatial redundancy techniques, the area and power overhead is high. Therefore aerospace circuits would benefit from adopting temporal redundancy instead, but existing solutions prioritize performance over reliability. Our proposed temporal redundancy latch-based architecture (TRLA) is a standard cell, static CMOS temporal redundancy technique, with area savings of 26%, power savings of 46%, and 14% faster circuit operation compared to triple modular redundancy (TMR).**

I. Introduction

Device shrinking is accompanied by smaller supply voltages and circuit node capacitances, reducing noise margins and thus reliability [13]. As a result, charge stored on circuit nodes decreases, increasing the sensitivity to charged particles traveling along the nodes [12]. Therefore ignoring the environmental influence on integrated circuits, on ground level and especially for aerospace electronics [14], is not feasible anymore. A particular malfunction caused by energetic particle strikes are current transients. Such single-event transients (SETs) may corrupt the state of a latch if they happen at the feedback node, or if they are propagated from upstream logic and sampled.

Although it is possible to mitigate SETs with spatial redundancy techniques such as modular redundancy and voting, the area and power overhead is high. A temporal redundancy technique which supports retrofitting of existing designs for soft error mitigation is not available. Our proposed TRLA is a standard cell temporal redundancy technique for aerospace applications, with speed, area, and power benefits compared to TMR. The key idea of TRLA is to monitor the output node of a conventional latch for invalid signal transitions and to correct errors on the architectural level. This paper contributes the following results:

1) A new algorithm for the local mitigation of SETs aware of critical error states for correction on a global level (Section III-B1).

2) A new controller and architecture with multiple-bit upset (MBU) mitigation capabilities implementing our proposed error correction algorithm (Section III-B2) which is able to handle the same amount of errors as a reference TMR implementation (Section IV).

3) Quantitative area, speed, and power results for a commercial 90 nm standard cell library with area savings of 26 %, power savings of 46 %, while being 14 % faster compared to a reference TMR implementation (Section IV).

II. Related work

Several error detection sequential (EDS) circuits have been proposed, which rely either on transition detectors, supply rail monitors, or double sampling with comparison to detect late signals. The Razor2 latch [7] is augmented with a delay chain and rising edge enabled dynamic transition detector to detect late signals. Transition detector with time borrowing also employs a transition detector with dynamic evaluation, whereas double sampling with time borrowing (DSTB), Bubble Razor, and TIMBER latches use a double sampling and comparison strategy [4, 6, 9]. The Razor Lite flip-flop [10] monitors virtual supply rails, stemming from the dynamic design style, to detect invalid signal transitions after the sampling edge.

Most EDS circuits resort to a dynamic CMOS design style, instead of more noise-robust [11] static CMOS, which is an adoption burden, especially in the aerospace context where the trend is towards commercial off-the-shelf FPGA implementations [1, 5, 8]. Double sampling and comparison for error detection is not suited due to metastability problems [2] arising from frequent signal transitions around the sampling edge.

Because manual (re)design for EDSs is inefficient and error-prone, transparent electronic design automation supported solutions are preferred. The elegant local error handling and transparent EDS retrofitting in Bubble Razor [9] is therefore favored over DSTB [4] or Razor2 [7], but it lacks SET mitigation capabilities. We fill this gap with our static, standard cell based EDS with SET mitigation capabilities presented in Section III.

While error detection is handled by the EDS, error correction can be accomplished by repetition of erroneous computations. With a dual-phase latch-based data path it is possible to delay the sampling of error-flagged data at the receiving latch until the correct result from the repeated computation is available.

Delayed sampling for local error correction was successfully demonstrated for *timing errors* by the Bubble Razor algorithm [9], which *assumes that invalid signal transitions indicate late but nevertheless correct signal values*. Our proposed algorithm in Section III-B1 *drops the assumption of correct late signals* to handle late and invalid signal values, *expanding the correction capabilities* to errors stemming from SETs.

III. Mitigating soft errors

In our temporal redundancy latch-based architecture the problem of error detection and correction is split up between the novel static CMOS EDS, presented in Section III-A, and the

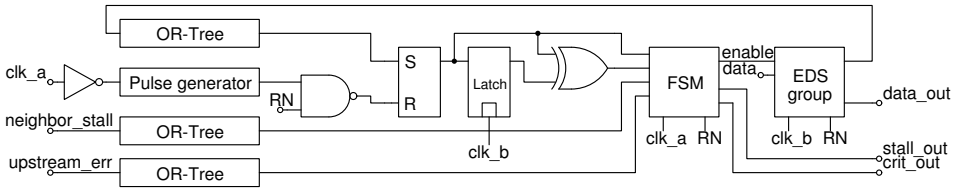


Figure 1. TRLA cluster architecture.

architecture for local error correction in Section III-B which implements our algorithm for SET mitigation.

The general idea of TRLA is to replace each unprotected latch in the design with our EDS to detect errors. The error information of each EDS is gathered and evaluated by logic that controls an enable signal for each latch to correct errors by recomputation.

A. Error detection

Our TRLA EDS resorts to a *transition detector* to monitor the output node of a conventional latch, as displayed in the upper right corner of Fig. 2. Late signal transitions during the transparent phase of the latch excite the transition detector which generates a pulse response that is captured by an SR-latch. The SR-latch is protected against single-event upsets (SEUs) by triplication and majority voting. The area requirements are alleviated by sharing of the protected SR-latch between multiple EDSs. To enable correction by recomputation, each EDS multiplexes between its input data and feedback from the output node, as shown in the upper left corner of Fig. 2. An SET at the output node close to the sampling edge is flagged by the *late error detector* in the bottom right corner of Fig. 2 and requires special treatment due to the propagation delay of the error correction logic.

The EDS is not exposed to the increased metastability risks described earlier, because it samples on the falling edge, which is not exposed to frequent data signal changes since the propagation delay t_{pd} of all paths feeding into the EDS are sufficiently short by design: $t_{pd} \leq T - t_{cq}$ with T as the cycle time and the clock-to-q delay of the latch t_{cq}. Compared to most EDSs heritage in timing speculative designs, where late signals are frequent by design and increase the risk for metastability [2], TRLA does not expose the sampling edge to frequent changes and therefore avoids a mean time between failures (MTBF) degradation.

The focus of this work is on mitigation of single-event effects (SEEs), therefore the pulse gating of the SR-latch is kept to a minimum, instead of increased pulse-widths leading to controlled time-borrowing. In this setup our EDS operates like a pulsed latch without further time-borrowing.

B. Error correction

While error detection is handled by the EDS, error correction is accomplished by repetition of erroneous computations. Due to the dual-phase latch-based data path it is possible to delay the sampling of error-flagged data at the receiving latch until the correct result from the repeated computation is available. Our algorithm in Section III-B1 describes the behavior of a distributed controller responsible for controlling each EDS enable signal to implement the correction by recomputation.

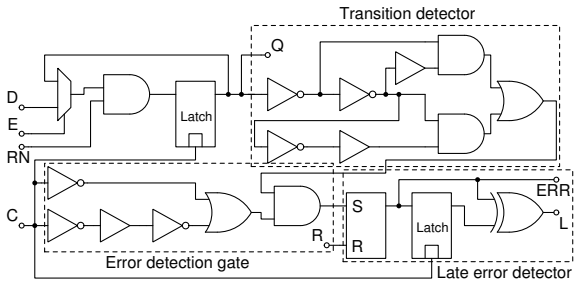


Figure 2. Static CMOS EDS with transition detector, late error detector, and optional gating for time-borrowing.

This controller is implemented as a finite state machine (FSM) alongside the EDSs and additional error signal evaluation logic, forming a cluster, as described in Section III-B2. Clusters group EDSs with a controller; the smallest possible cluster combines only one EDS and controller. They abstract an arbitrary number of latches in the described algorithm.

1) Algorithm: If a cluster of several EDSs, which operate on the same phase, disable sampling, this cluster of EDSs stalls. Due to the dual-phase clocking scheme, the minimum number of clusters is two. Clusters are connected with other clusters or primary in- and outputs, inheriting the flow of data from the unprotected source design. Thus one needs to differentiate between downstream clusters, which are neighbor clusters in direction of the data flow, and upstream clusters, as neighbors in reverse data flow direction.

Our algorithm evaluates error conditions within its own and in neighbor clusters, to decide about stalling of the EDS in its own cluster. This stalling scheme, as shown in Fig. 3 for two clusters is needed for successful preservation of valid signals and avoidance of sampling invalid ones. In Fig. 3, an invalid signal is sampled and then corrected by recomputation, followed by detection of a late error in the shaded region which is rated critical by the controller. Dotted clock pulses indicate that the corresponding EDSs do not sample new data. Arrows are used to show the exchange of error information and to highlight where stalling decisions are taken. Local error and late error are aggregated signals from all EDSs nodes labeled “ERR” and “L” in Fig. 2. The algorithm in Fig. 4 describes the behavior of the system, consisting of multiple interconnected FSMs in context of the architecture, described in Section III-B2, to correct errors.

Compared to Bubble Razor, which assumes late but nevertheless correct signal values, our algorithm interprets late signals as manifestations of SETs, and ensures that the circuit state is not corrupted. The lean implementation of Bubble Razor’s control logic, which merges error signals from the data path into the evaluation of the exchanged stalling information [9], is possible due to the restriction to timing errors. Thus local soft error

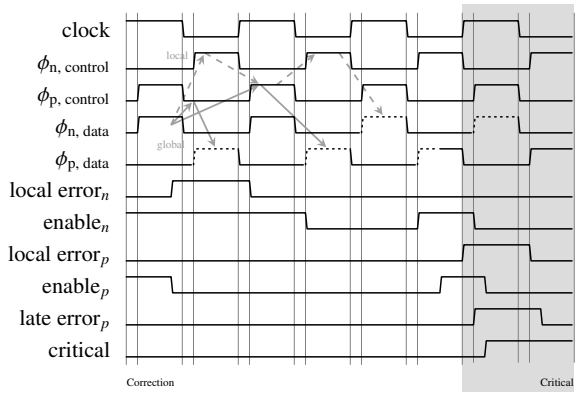


Figure 3. Stalling scheme for two clusters.

```
1: function CRITICAL(error information)
2:     c ← late error detected
3:     c ← c ∨ local error during upstream error
4:     return c
5: end function
6: procedure CORRECTION(state, error information)
7:     if ¬CRITICAL(error information) then
8:         for all cluster do                          ▷ in parallel
9:             if local error then
10:                Stall neighbors to recompute
11:                Sample recomputed value
12:                repeat
13:                    Stall local EDSs
14:                until Downstream correctly samples
15:            end if
16:        end for
17:    else
18:        Escalate error to system level
19:        Disable all EDSs for possible state recovery
20:    end if
21: end procedure
```

Figure 4. Correction algorithm

correction by recomputation is not possible for Bubble Razor. Contrary our algorithms accounts for the correction of soft errors in a conflict-aware manner, which requires a differentiation between errors and stalling information.

2) Distributed implementation: We implement our algorithm using FSMs responsible for exchanging error information between neighbor clusters and control of the data path latch enable signal. Each cluster resorts to the same FSM design but operates it on the clock phase complementary to its neighbor clusters. The state machine in Fig. 5b evaluates four sources of information: *1)* Stalling information from all neighbors i_1; *2)* Detected errors in clusters which are part of the upstream from a data path perspective i_2; *3)* Locally detected errors which happened in the cluster controlled by the FSM i_3; and *4)* Locally detected late errors. The FSM generates three signals for latch control and communication of errors: *1)* Stalling information sent to up- and downstream neighbors o_1; *2)* Local EDSs enable signal o_2; and *3)* Critical state reached, local correction not possible o_3. Furthermore, local errors detected by the EDSs are always communicated to the downstream neighbors regardless of FSM state. Locally detected late errors always transfer the FSM into the critical state S_c. The corresponding edges are excluded from Fig. 5b for clarity. The mealy automaton starts in the idle state S_i, and given a local error the automaton transitions to the long

Table I
RELATIVE COMPARISON TO THE UNPROTECTED DUAL-PHASE LATCH-BASED DESIGN.

Design	$A/\mu m^2$	f_{max}/MHz	A_{rel}	$f_{max,rel}$
Unprotected	21579.4	168.1	1.00	1.00
TRLA	71414.5	156.3	3.31	0.93
TMR	96567.0	137.0	4.48	0.82

suppressing state S_l followed by the resume state S_r. The short or quick suppressing state S_s is reached if neighbors are stalling or upstream cluster errors are detected. The critical state S_c is entered once error conditions arise which are not locally correctable by the automaton.

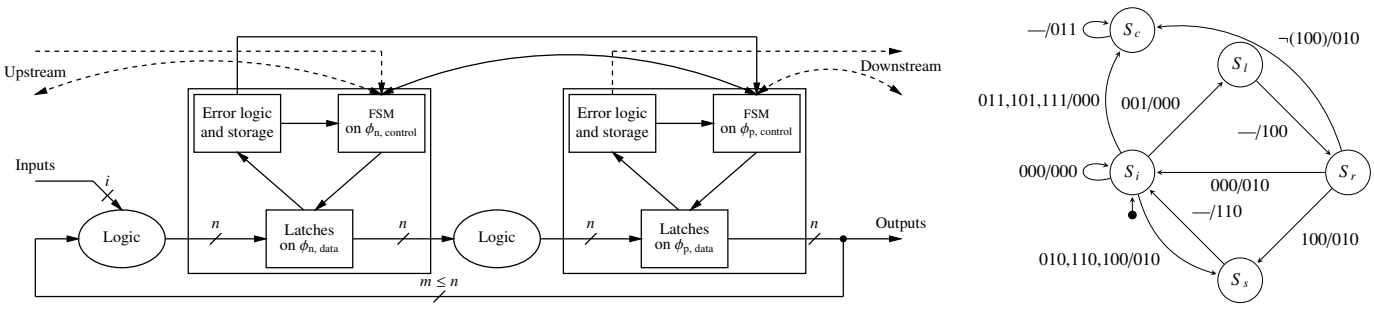
Our cluster architecture determines the information sources of each FSM, arranging them to perform according to our algorithm. The cluster architecture is shown in Fig. 1 and allows to share components between our EDS to save area. Each cluster has a polarity stemming from data path clock phase requirements. The pictured cluster is of negative polarity with respect to the clock of the original synchronous design. The EDSs share a single pulse generator for the error detection gate and a single error signal storage SR-latch. The output of the SR-Latch is sampled by a level-sensitive latch operating on the same phase as the data path latches, to preserve the error information and to detect late errors. An additional pulse generator periodically resets the error signal storage SR-latch on the falling clock edge of the phase the cluster control logic operates on. This erases outdated error information prior to the next transparent data path period, implicitly allowing arrival of new data during the opaque phase of the data path latches.

All stateful elements in the control path of our architecture resort to triplication and majority voting to prevent MTBF degradation due to SEUs. The area overhead is benign because the state of the control path is only captured in the error information latches and the state vector of the controller FSM.

IV. EXPERIMENTAL RESULTS

The TMR and TRLA-protected designs compared in Table I are protected versions of the same unprotected J1 CPU design [3]. The TMR contains complete triplication of the fan-in logic, latches, and redundant voting. The standard cell library used for synthesis contains specialized majority voting cells. Still the TRLA solution requires 26 % less area compared to TMR. In contrast to the sequential error correction in TMR our TRLA protection mechanism operates in parallel to the actual data path and allows to operate the circuit 14 % faster.

To estimate the impact of the circuit modifications, gate-level simulations with back annotated timing information are augmented with a simulated radiation source. The radiation source is written as a script that traverses the circuit to find all nets in the fan-in of latches. For a requested number of particles a random signal level (0 or 1) is placed on nets uniformly chosen in space and time, simulating an SET, resulting in different combinations of SETs, SEUs, and MBU. During reset of the circuit the radiation source is disabled. Once the circuit is in a known state, the irradiation starts while the CPU endures a heavy load on the arithmetic, control and memory system to maximize



(a) Protected circuit with controlled data path latches (b) Mealy automaton

Figure 5. Block diagram of circuit protected by TRLA and error correction automaton.

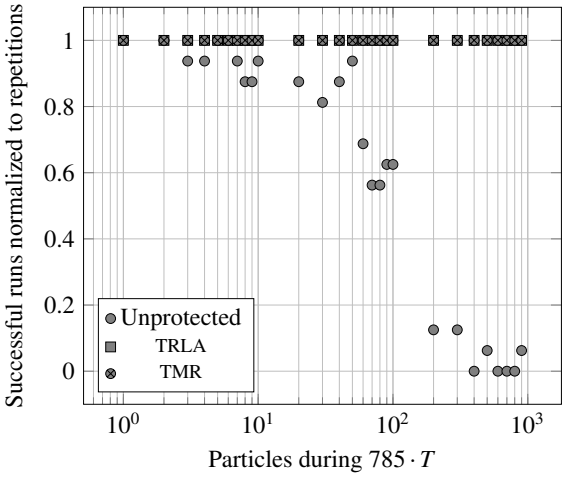


Figure 6. Number of program executions without deviations over number of radiated particles.

Table II
SWITCHING ACTIVITY BASED POWER CONSUMPTION ESTIMATES FOR RUNS IN FIG. 6.

Design	Total power P/mW	Relative increase
Unprotected	2.9577 ± 0.0239	1.00
TRLA	3.1992 ± 0.1478	1.08
TMR	5.9755 ± 0.0005	2.02

the impact of each fault on the observable outputs. All signal switching activities are recorded and used for power estimation with information from the standard cell library using industry grade tools.

The simulations are repeated for each circuit to gather a dataset for all chosen numbers of radiated particles that cause an SEE during program execution. Fig. 6 shows the number of successful program executions for each amount of radiated particles. While the unprotected design starts to decay at the presence of several particles, both TMR and TRLA ensure correct operation. The consumed power estimates in Table II are the arithmetic mean over all simulation runs. For the TMR protected design the consumed power is nearly static with only small deviations due to the regular voting. In contrast our TRLA error correction operates on demand, resulting in more deviations and less total power consumed.

V. CONCLUSION

Our temporal redundancy latch-based architecture (TRLA) protects arbitrary single-clock synchronous digital circuits. Compared to triple modular redundancy and voting, we save 26 % area, 46 % in consumed power, while being able to operate 14 % faster. Gate-level simulations are used to show the mitigation capabilities of the architecture, with reports of complete error mitigation in all runs conducted even for artificial high fluence rates. The presented toolflow allows to retrofit designs without designer intervention, making TRLA a prime candidate for retrofitting of temporal redundancy for already existing designs resorting exclusively to static CMOS circuits.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 637616.

REFERENCES

[1] S. Avramenko et al. On the Robustness of DCT-based Compression Algorithms for Space Applications. *IEEE Int. Symp. on On-Line Testing and Robust Syst. Design*. (2016).
[2] S. Beer et al. Metastability in Better-Than-Worst-Case Designs. *20th IEEE Int. Symp. on Asynchronous Circuits and Syst.* (2014), p. 101–102.
[3] J. Bowman. J1: a small Forth CPU Core for FPGAs. *26th EuroForth Conf.* (2010), p. 43–46.
[4] K. A. Bowman et al. (2009). Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. *IEEE J. Solid-State Circuits* 44(1):49–63.
[5] S. Campagna et al. A framework to support the design of COTS-based reliable space computers for on-board data handling. *16th IEEE Int. On-Line Testing Symp.* (2010), p. 91–96.
[6] M. R. Choudhury et al. (2014). Time-Borrowing Circuit Designs and Hardware Prototyping for Timing Error Resilience. *IEEE Trans. Comput.* 63(2):497–509.
[7] S. Das et al. (2009). RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance. *IEEE J. Solid-State Circuits* 44(1):32–48.
[8] S. Esposito et al. (2015). COTS-Based High-Performance Computing for Space Applications. *IEEE Trans. Nucl. Sci.* 62(6):2687–2694.
[9] M. Fojtik et al. (2013). Bubble Razor: Eliminating Timing Margins in an ARM Cortex-M3 Processor in 45 nm CMOS Using Architecturally Independent Error Detection and Correction. *IEEE J. Solid-State Circuits* 48(1):66–81.
[10] I. Kwon et al. (2014). Razor-Lite: A Light-Weight Register for Error Detection by Observing Virtual Supply Rails. *IEEE J. Solid-State Circuits* 49(9).
[11] P. Larsson et al. (1994). Noise in Digital Dynamic CMOS Circuits. *IEEE J. Solid-State Circuits* 29(6):655–662.
[12] M. Nicolaidis (2005). Design for Soft Error Mitigation. *IEEE Trans. Device Mater. Rel.* 5(3):405–418.
[13] M. Nicolaidis. Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies. *17th IEEE VLSI Test Symp.* (1999), p. 86–94.
[14] *Space engineering: Space environment*. Standard ECSS-E-ST-10-04C. ESA Requirements and Standards Divison, 2008.