SET Hardened Derivatives of QDI Buffer Template

Zaheer Tabassam, Andreas Steininger
Institute for Computer Engineering, TU Wien, Vienna, Austria
zaheer.tabassam@tuwien.ac.at, steininger@ecs.tuwien.ac.at

***Abstract*—As critical charges become smaller due to technology advancement, Single Event Transients (SET's) become more threatening to circuits. Quasi Delay-Insensitive (QDI) circuits are tolerant against timing issues, but they tend to be more sensitive towards transients – and hence SET's – in the value domain. This can be somewhat mitigated, without sacrificing their delay insensitivity, by shortening their sensitive data acceptance windows.**

In this paper we investigate these sensitive areas in search of possible ways to specifically harden buffer stages, as these are elementary for building asynchronous pipelines and play a major role in the manifestation of an SET as a Single Event Upset (SEU). Inspired from existing work in the literature, we propose a buffer template called "Dual CD IN/OUT Interlock WCHB" that is basically a hybrid approach to smartly shorten the sensitive window. It reduces the cases where existing approaches fail by up to 5% magnitude. Further investigation suggests some improvement in the design, namely the "Dual CD IN/OUT Interlock WCHB Simplified" which leads to up to 44% area savings without effecting the core resilience.

The enhancements are verified in simulation with realistic circuits like Multiplier, ALU, and FIFO under a timing model from the NanGate 15nm library.

I. Introduction

As delays continue to become increasingly unpredictable and variable in modern VLSI technologies, the worst-case delay assumption underlying the synchronous design paradigm is getting cumbersome and inefficient. Consequently, asynchronous design techniques, whose closed-loop timing design can flexibly adapt to delay variations, are considered an attractive alternative. As a further benefit, this flexible timing makes asynchronous circuits, most notably the so-called QDI ones, also robust against fault effects that impact the circuit's temporal behavior. Due to their event-based operation principle, however, QDI circuits are deemed quite sensitive to transient faults in the value domain, like glitches. The latter are often encountered as SET's that result from ionizing particles hitting a transistor junction. Unfortunately, the rate of SET's is also increasing, as feature sizes shrink in modern technologies. Hence, making QDI circuits more resilient against SET's becomes a very relevant issue.

The wealth of fault-tolerance techniques available for synchronous circuits can, unfortunately, rarely be directly applied to QDI circuits, due to their different operation principle. The few hardening techniques that have been proposed specifically for QDI circuits still have some shortcomings. In this paper we will closely investigate these and, based on the insights thus gained, propose three improved approaches. Our focus herein will lie on pipeline buffer stages, as these are elementary generic building blocks for asynchronous designs and play a key role in converting transient pulses into (lasting) state changes. We will compare the robustness of different buffer designs by means of extensive fault-injection experiments in simulation and thus give evidence for the benefits of our approaches. We will furthermore show that our techniques are very efficient by incurring only very moderate area and performance penalties.

This research was partially supported by the project ENROL (grant I 3485-N31) of the Austrian Science Fund (FWF).

II. Related Work

The focus of this work is on the mitigation of transient faults in asynchronous circuits, specific to the value domain, so here we survey models, effects and hardening techniques for dealing with them. In synchronous systems transient faults are efficiently mitigated through masking capabilities that are partly inherent, and, where required, additionally established with fault-tolerance techniques. The latter, however, tend to have large overhead and architecture constraints when ported to asynchronous systems.

Over the years, researchers have regarded redundancy as key requirement for error resolution and taken inspiration from synchronous hardening methods for protecting asynchronous circuits, like [1], [2]. This strategy is backed by promising results for duplication-based approaches [3], [4]. By leveraging this topology [5], [6] and [7] proved the resilience of real-word asynchronous circuits like processors and controllers. In addition, [8], [9] and [10] highlight the main contributors that must be considered for Single Event Transient (SET)-tolerant asynchronous circuits. Investigations and results of [11], [12], and [13] suggest special ways to apply this redundancy to asynchronous circuits.

Inspired from [4], [13], and [14], the authors of [15] proposed two enhanced QDI buffer styles with high resilience for their specific domains while maintaining low area overhead. As a continuation to this, with a real-word circuit "multiplier", [16] presents a more elaborated view of QDI templates with respect to transient faults. They not only compare templates, but also illustrate how their behavior varies with circuit dynamics. Based on [15], [17] presents more robust QDI buffer templates that shorten the windows where SET's are harmful.

In conclusion, mere replication of a circuit can enhance SET tolerance, but for a high area cost. Consequently, it seems

promising to rather analyze the key contributors to faults in order to selectively target smaller modifications.

III. Asynchronous Logic

Where in synchronous circuits a clock provides global temporal coordination within the circuit, in asynchronous circuits each module maintains local synchrony with its neighbors via handshakes. The handshake cycle of circuits realized with a Delay-Insensitive (DI) protocol is more flexible in terms of timing assumptions, because the validity of data is defined by the data itself using multi-rail encoding schemes. An explicit $acknowledgement$ signal from the receiver completes the handshake cycle [18].

From the choice of available options we focus on QDI circuits throughout this article[1], and we stick with a *4-phase return-to-zero* handshake protocol with Dual-Rail (DR) encoding. In the DR scheme a single bit x is represented by two rails $(x.t, x.f)$ where t and f are $true$ and $false$ rails, respectively [18]. A $logical$"1" is represented by setting these rails to $(1, 0)$ and "0" by $(0, 1)$. These are code words and called *(data) tokens*. The code $(0, 0)$ is used as a *spacer*, as demanded in the *4-phase protocol* to separate *data tokens*. Note that for the considered scheme $(1, 1)$ is an illegal pattern. In this protocol without global clock a module interacts with an other by simply placing a *data token* on its data rails, and after receiving a logical high $acknowledgement$ signal from the receiver it changes the *data token* for a *spacer*. The handshake is completed with its 4^{th} phase when the receiver responds by resetting the $acknowledgment$ [18].

A. The Muller $C-element$

The main functionality of a Muller C-element (MCE) is simple: only if all inputs have the same logic level (switching condition), that level is passed to the output (after a propagation delay) and retains saved until a matching pattern with the opposite logic level arrives at the inputs. There are asymmetric C-elements that obey extra conditions on special inputs. Figure 1 CT-part illustrates these behaviors.

(a) MCE with negative input (C-element-): For up-transitions the input NegIN has no impact on the switching condition, see up-transitions on IN3 and IN4, triggering a transition on Out2. To change Out2 to logical "0", however, all inputs, including NegIN, must be set to "0". This is not yet the case with transitions (1) and (2), but happens after (3), which triggers (4) at Out2.

(b) MCE with positive input (C-element+): For the up-transition (8) at Out3 the normal inputs IN5 and IN6 must be high ((5) and (6)). However, (7) at PosIN is also required. In contrast, (11) at Out3 only requires (9) and (10) because for the down-transition the C-element+ ignores PosIN.

Note that whenever a switching condition is fulfilled, the state (output) of the MCE is determined by the inputs alone; this is called the *combinational mode* of operation. In contrast, otherwise the internal storage cell alone determines the output; this is called the *storage mode*. Figure 2 shows a symbolic representation of a MCE and its variants.

[1]The key difference to DI is the isochronic fork assumption constraint [19]

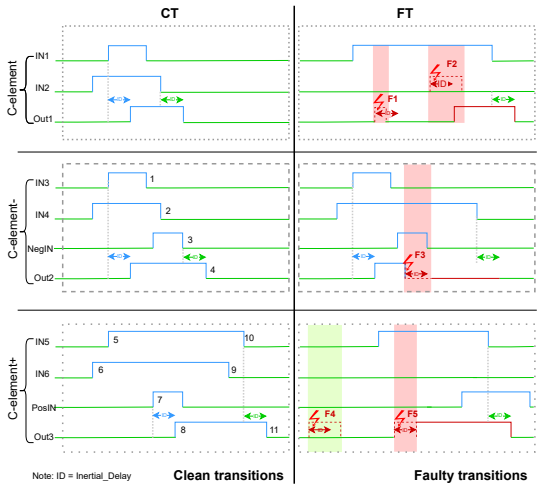

Fig. 1. MCE behavior under normal and faulty conditions

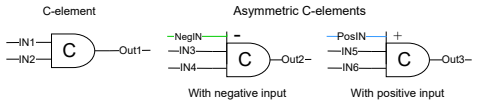

Fig. 2. MCE and its derivatives

B. QDI Buffer Templates

As QDI circuits operate with handshakes rather than a global clock, they require special storage elements that also obey the handshake protocol with the respective encoding scheme. In this paper we will restrict ourselves to the popular 4-phase QDI buffer template named Weak-Conditioned Half Buffer (WCHB). We will start with the basic template and then continue with some of its variants that are designed to mitigate the effects of SET's as shown in Fig. 3. To save space we integrate two buffer styles in one package, the base buffer and its Dual Completion Detection (DCD) variant. The orange part highlights the DCD part. In this section, we only consider a 1-bit WCHB (α without orange part, enable is directly connected with the inverted acknowledgment line): according to the DR scheme, only one input rail may go high at a time, and if the $enable$ (en) signal is also high, the respective MCE fires, which, in turn, activates the $acknowledgment$ Ack_out. When the MCE receives the Ack_In from its successor (in this scenario the sink) it gets armed for capturing the spacer. If due to a fault both input rails (In.T, In.F) go high while (en) is high, the illegal (1,1) pattern propagates to the output.

C. Pipeline Load Factor

QDI circuits flexibly adapt their operation to the speed in which tokens are provided by the source and consumed by the sink. To illustrate that, we are referring to the WCHB from Fig. 3 and its waveform in Fig. 4. In BOX-A, e.g., a token is

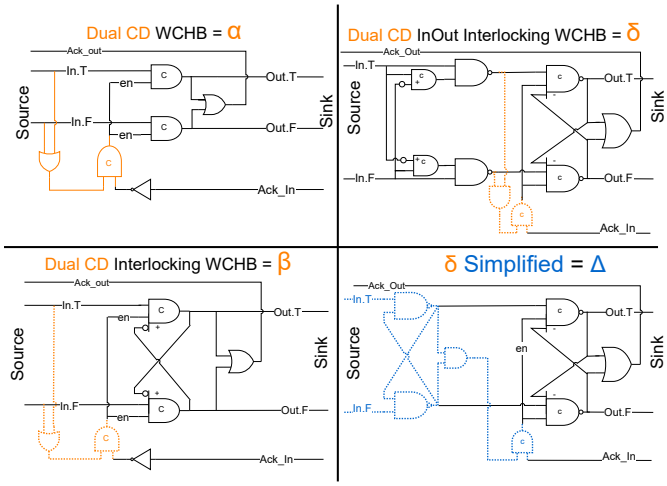

Fig. 3. Considered Buffer Styles

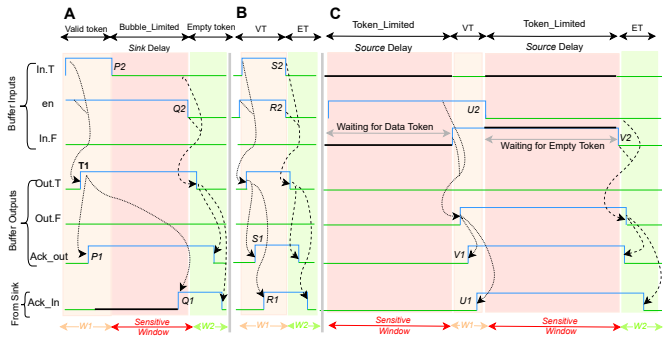

Fig. 4. Waveform of Fig. 3 (WCHB) with Bubble and Token Limited modes

provided to Sink at T1, to which the latter responds after some delay at Q1, highlighted as "Sink Delay", while a new empty token is already waiting at buffer input (P2) to be latched. This behavior is called $Bubble\_limited$ because $Sink$ is slow as compared to $Source$, so most of the time the pipeline is waiting for $acknowledgment$ (termed "bubble", as a counterpart to "token") to complete the handshake cycle. In contrast to that, in BOX-C $Source$ is slow where buffer is waiting for a new data token or spacer (empty token) highlighted as "Source Delay". $Sink$ responds to the data token at "U1". The latter ripples to en and arms the buffer for spacer at "U2" where "V1" is the corresponding request. $Source$ responds at "V2" after some delay. This case where the pipeline is stalled by the $Sourcedelay$ is called $Token\_limited$ mode of operation. The Pipeline Load Factor (PLF) has also been used by [16] to express the degree to which tokens or bubbles limit the pipeline speed. For the remaining discussion a PLF less than 1 indicates a $Token\_limited$ mode of operation, and a PLF greater than 1 a $Bubble\_limited$ mode. With the QDI design style, the PLF, while affecting the throughput, has no impact on data integrity. In context with transient faults, however, the PLF makes a difference. Therefore we will consider it in the following.

D. Error Types in QDI Circuits

Deviations from the correct behavior (as recorded in a golden run) are classified into two main categories, namely data errors and timing issues. For timing issues, i.e., data arriving earlier or later than expected, data integrity remains safe, because of the circuit's delay insensitivity. Data errors are further classified into four categories [16].

(a) Value error: data is received correctly, but the value is not as expected.
(b) Coding error: both rails of the DR bit go high; this is illegal in our considered DR encoding.
(c) Glitch: during any handshake phase a signal makes more than one transition, or causality of signals is otherwise violated by a wrong sequence, like $acknowledgment$ activated before data completion.
(d) Deadlock: the circuit stops in a state where no further transition is possible.

In an earlier analysis we could establish that glitches do not propagate as such in an asynchronous circuit; they only become observable when directly affecting the control signal of the first or the last stage in a pipeline [20]. Even in those cases they always triggered other types of data errors. So in our analysis we exclude glitches and reduce the considered types of data errors to the remaining 3 effects.

IV. Threats to the QDI Pipelines

In this section we identify, in theory, the main contributors and reasons where QDI circuits lose their resilience to SET's.

A. Sensitive Areas of the $C-element$

As QDI circuits are mainly composed of MCEs, these have a major contribution in error generation and propagation, but also masking. The right part of Fig. 1 (labelled "FT") illustrates possible effects on the three MCE types under fault scenarios. These are numbered F1 to F5. Discussing all fault combinations is not feasible here – the purpose of the following discussion is to show some principal effects.

1) F1: A fault pulse hits the MCE output Out1 while in storage mode (non-matching inputs). In the physical implementation this fault would attempt to flip the MCE's storage loop from the output side, but the one shown here is too short to overcome the element's inertia.
2) F2: The MCE input rail In2 is hit by a fault. While it persists, the inputs match. Since this time the fault pulse is sufficiently long, Out1 is indeed flipped and remains at the erroneous level even after the fault vanished.
3) F3: A negative fault pulse forces Out2 to "0". Note that right before the fault occurred, the C-element- moved to storage mode, by a transition on IN3. However, in contrast to F1, this fault is long enough to flip the state.
4) F4: A positive fault pulse affects Out3 of C-element+. Even though its length is sufficient, it has no effect because the MCE is in combinational mode.
5) F5: A positive fault pulse hits Out3 while in storage mode. Its length is sufficient to flip the state – even though PosIN is at LO.

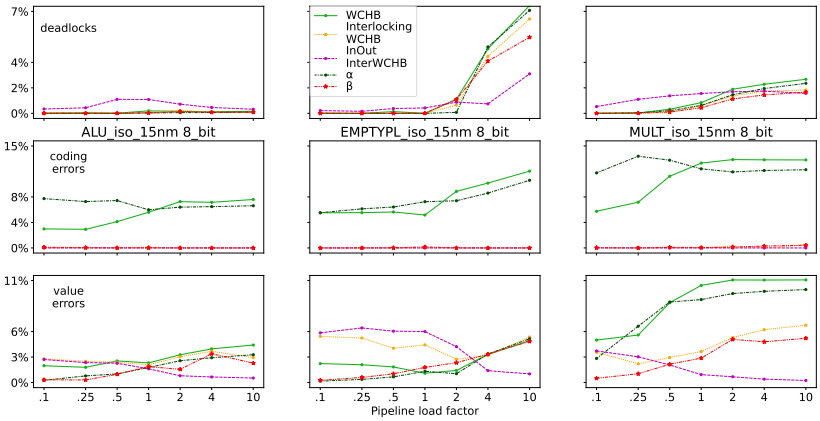

Fig. 5. Error rates of α, β and their base templates

B. Sensitivity of Different Buffer Templates

In IV-A we have elaborated that the MCE is sensitive to SET's while in storage mode, i.e., when its inputs levels do not (all) match. We have further seen in III-C that in pipeline operation a WCHB captures data when both, data and en are present (matching). In token-limited mode en is already present, but data needs to arrive. During that waiting period the MCE is in hold mode and its data input as well as its output sensitive to SET's. A similar sensitivity window emerges in bubble-limited mode: While data is already there but en still missing, the MCE's en input and data output are sensitive. Figure 4 indicates sensitive windows by a red shading and insensitive ones (combinational mode) in green. Note that in the DR encoding only one of the two data rails is supposed to switch. So while one of the MCEs already switched, the other one still remains sensitive until its en is removed. These phases are indicated in Fig. 4 as "W1".

After thorough analysis of sensitive windows of the WCHB, [15] proposed the "Interlocking WCHB" (indicated as β) shown in Fig. 3 without the orange part. Here the MCE whose output transitioned to "1" first, blocks the other MCE from making the same transition, which effectively prevents a (1,1) output and closes the data accepting window. Due to the inertial delays of the MCEs, however, this interlocking loses its effectiveness when a fault hits the input signal before the correct transition, gets latched and then blocks the correct transition arriving on the other rail. So this method is only effective for a few scenarios when the fault hits after the correct transition happened, with the conditions that (a) the buffer is already in a locked state and (b) the fault must not hit from the output side.

To address this issue [17] proposed the InOut Interlocking WCHB (IOIWCHB) shown in Fig. 3 (δ, without orange part). This approach provides a first stage of interlocking (actually input filtering) that just compares the inputs without considering the output and hence does not suffer from the delay problem as the Interlocking WCHB does. This stage is followed by a second one that actually resembles an Interlocking WCHB, albeit with inverted inputs (this is done to optimize area). This latter stage protects against faults at the output that might make the state flip to (1,1). In addition, the NAND gates with their delayed and non-delayed version of the same signal provide some degree of glitch filtering.

V. Experiment Setup

Different circuits have their inherent properties and behave differently under different conditions. To explore some of these variants we chose fundamental but useful real-world circuits for our investigations. Our list comprises Pipelined Multiplier, Arithmetic Logic Unit (ALU), and Empty Pipeline or FIFO. The data bit width remains same for all, namely 8-bit.

Our simulation setup consists of a QDI source and sink generating and $acknowledging$ DR data with programmable delays to mimic real-world DI scenarios. With these programmable source/sink delays we control the PLF. Monitors are placed at the interfaces of the target circuit that check each activity and compare it to a golden run. Please note that only value deviations are considered as error, while timing issues are only considered an observation because circuits are QDI.

For three target circuits with seven different buffer styles, each with 7 different PLF choices, we got a total of 147 circuit variants, for which we ran 5795148 simulations. As our concern is SETs, which occur rarely, we only inject one fault per simulation and observe the behavior. We are excluding input and output signals from the injection list because these are directly observable to the monitors with no chance for mitigation or masking, so these will simply result in higher fault rates. Injection time and location are randomly chosen, while the injection pulse length is fixed to a value higher than the longest inertial delay among all gate delays in our considered buffer templates, and hence electrical masking cannot occur (we are only interested in logical and temporal masking effects here). To run the simulation we use a network of 10 physical machines (3.5 GHz 7 th generation Intel i5 processor, 16 GBRAM) each running 4 workers in parallel (one worker per core). The actual simulator used is QuestaSim (version 10.6c). After all simulations are complete the final results can be extracted from the database using SQL queries.

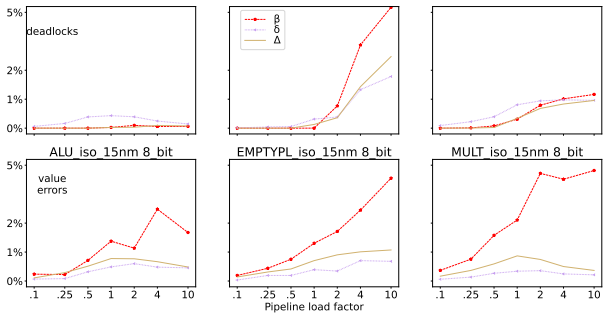

Fig. 6. Overall comparison between proposed approaches

For the gate delays we utilized timings from the NanGate_15nm library file with typical conditions [21]. As an MCE is not part of this library, we considered the MCE model from [22] in which they propose a combination of simple NAND gates. Gate delays are computed from a timing matrix of the respective gate using an interpolation method with fixed index_1 (input_net_transition), as in our simulations we are not varying this parameter, while we vary index_2 (total_output_net_capacitance) depending on the fanout of the respective gate.

VI. Enhancing the Fault Tolerance

A. Possible Solutions

As evidenced by case F4 in Fig. 1 and our analysis in IV-B the MCE is immune to SET's at its output while data and en match (green shaded areas in Fig. 1, termed "W2"). In conclusion, faults appearing at the buffer output are easily addressed by maintaining the MCE inputs until both source and sink responded to the last respective token. In this window the circuit is also resilient towards SET's at the input. [13] already proposed the idea called "normally closed pipeline latch" where the buffer only passes the ack (or en signal) to the MCE after validating the respective new token. In the following we will refer to that technique as "Dual CD WCHB" or short α, see Fig. 3.

1) Token limited mode: From a theoretical perspective the α approach seems very robust for part-C ("Sensitive Window") of Fig. 4. However, experiments as presented in Fig. 5 unveil that the basic WCHB as well as α (with our reference) has no resilience towards coding errors because even α only protects the MCE that is going to make transition during the respective handshake cycle. For example, in Fig. 4 "B" during "W1", when In.T and en are high, the true rail MCE is holding a logical "1" but the false rail MCE with In.F is still armed for firing due to high en signal. If a fault hits In.F the respective MCE generates a logical "1" which results in an illegal (In.T = 1, In.F = 1) code word. [15] answer this coding error issue of the WCHB by simply locking the opposite rail MCE after firing one of them. However, this approach known as Interlocking WCHB simply considers the first rail to transition as the correct one and therefore comes with a price of more value errors at some places, as shown for "Empty Pipeline Circuit " in Fig. 5. As a first remedy we proposed approach β, with an additional DCD as shown in Fig. 3. β outperforms all others especially with the multiplier circuit. Statistics in Fig. 5 shows that the addition of the DCD protects the circuit by shortening the sensitive windows of Fig. 4 part-C. β resolves up to 6% value errors as compared to its base buffer as Fig. 5 "Empty_pipeline" suggests under token limited mode.

2) Bubble limited mode: When Source generates a new token (P2) before en makes its transition at "Q2", the α approach is not appropriate as now the data transition arms the MCE and moves it to storage mode, without the extra check by the additional CD becoming effective. One way to shorten this window is to add some extra delay to the data input. If this delay is added cleverly, it pays off in other regards as well. [17] proposed input interlocking to validate the token before passing it on to the buffer, which also provides glitch filtering property. This In/Out Interlocking WCHB is presented in Fig. 3 top right, without the orange colored AND gate and MCE. Our simulation results prove its resilience during bubble limited mode as shown in Fig. 5. At the same time they show that this buffer is more susceptible to faults during token limited mode because due to the gate delays the input interlocking mechanism increases the sensitive window.

B. Suggestions to achieve higher resilience

The results in Fig. 5 indicate that the proposed β approach is the best choice for token limited mode, where In/Out Interlocking WCHB from [17] excels in bubble limited mode. Here we propose δ as first improvement of [17] with the addition of a DCD after the input interlocking gates as shown in Fig. 3. This helps shortening the sensitive window again to approximately the same magnitude as β does. In this way δ, a hybrid approach proposal, performs well for all PLFs as shown in Fig. 6. The results confirm that we could safely shorten the sensitive windows presented in Fig. 4. As δ utilizes an MCE for input interlocking, these are also affected by all fault scenarios discussed in section IV-A. The biggest concern here are faulty transitions at the output, which, if they get latched, generate deadlocks. This is reflected by higher deadlock rates in Fig. 6. Consequently this approach is well suited when correctness is the main concern, rather than liveness. Enhanced versions limit the coding errors to 0%.

C. Further Enhancement for Area efficient Solution

After thorough analysis we finally came up with the Δ approach, a simplified version of δ as presented in Fig. 3. This simplified input interlocking mechanism inspired from [20] yields a reduction in area by approximately 44% and better throughput compared to δ, see Fig. 7. In the results from Fig. 6, Δ proves its error resilience compared to δ. The use of an SR latch for input interlocking instead of MCE as shown in Fig. 3 buys a reduced number of deadlocks through an increased rate of value errors. With the MUTEX-like input interlocking, the earlier transition wins (no matter whether it is the correct one) and then blocks the other one. If the SET is the winner, the result is a value error.

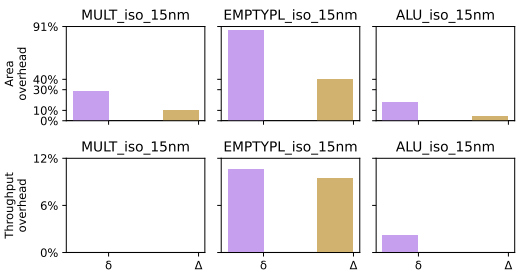

Fig. 7. Overall Area and Throughput Overhead Comparison with β as baseline

VII. Conclusion

In this work we first identify the main reasons behind the sensitivity of QDI buffer templates and then with real word circuits we run extensive fault injection simulations to validate and highlight the shortcomings of available approaches. We conclude that the $Dual\_Completion\_Detection$ approach in theory seems appropriate to shorten the sensitive window during $token\_limited$ mode but our experimental results shows that it converts value errors to coding errors with little improvement in $bubble\_limited$ mode. The higher coding error issue is resolved by proposing a hybrid approach: it utilizes the DCD to open the buffer and then locking it with the output interlocking technique after first transition is latched. With this technique we achieve higher resilience towards SET's in $token\_limited$ mode, but it lacks in $bubble\_limited$ mode.

From literature, the $Input/Output\_Interlocking$ technique shows promising results during $bubble\_limited$ mode as this technique leverages the benefits of temporal redundancy with an input interlocking mechanism. However, as this temporal redundancy widens the sensitive window, it yields worst resilience during token limited mode.

To address all delay scenarios we propose DCD variant of $Input/Output\_Interlocking$ called δ. Through careful placement of the input completion detection the sensitive window can be considerably shortened. This technique shows higher resilience compared to others but with a higher price in terms of area and deadlocks.

Further investigations finally lead to a more efficient solution called Δ. Without effecting the sensitive windows we saved up to 44% area by replacing MCEs used for input interlocking with an SR latch technique from literature. With no glitches and coding errors, δ and Δ restrict deadlocks and value errors to under 1%.

For future direction we are evaluating ideas to somehow manage delaying an incoming token to reach to buffer only when the enable signal is there during bubble limited mode. That would establish full resilience against SET's.

References

[1] T. Verdel and Y. Makris, "Duplication-based concurrent error detection in asynchronous circuits: shortcomings and remedies," in *Proceedings of the 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2002, pp. 345–353.

[2] F. A. Kuentzer and M. Krstic, "Soft Error Detection and Correction Architecture for Asynchronous Bundled Data Designs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–12, 2020.

[3] Y. Monnet, M. Renaudin, and R. Leveugle, "Hardening techniques against transient faults for asynchronous circuits," in *11th IEEE International On-Line Testing Symposium*, July 2005, pp. 129–134.

[4] W. Jang and A. J. Martin, "SEU-tolerant QDI circuits [quasi delay-insensitive asynchronous circuits]," in *11th IEEE International Symposium on Asynchronous Circuits and Systems*, March 2005, pp. 156–165.

[5] M. Marshall and G. Russell, "A Low Power Information Redundant Concurrent Error Detecting Asynchronous Processor," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, Aug 2007, pp. 649–656.

[6] S. Keller, A. J. Martin, and C. Moore, "DD1: A QDI, Radiation-Hard-by-Design, Near-Threshold 18uW/MIPS Microcontroller in 40nm Bulk CMOS," in *21st IEEE International Symposium on Asynchronous Circuits and Systems*, May 2015, pp. 37–44.

[7] F. A. Kuentzer, M. Herrera, O. Schrape, P. A. Beerel, and M. Krstic, "Radiation Hardened Click Controllers for Soft Error Resilient Asynchronous Architectures," in *26th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2020, pp. 78–85.

[8] Y. Monnet, M. Renaudin, and R. Leveugle, "Asynchronous circuits sensitivity to fault injection," in *10th IEEE International On-Line Testing Symposium*, July 2004, pp. 121–126.

[9] C. LaFrieda and R. Manohar, "Fault detection and isolation techniques for quasi delay-insensitive circuits," in *International Conference on Dependable Systems and Networks, 2004*, June 2004, pp. 41–50.

[10] R. P. Bastos, Y. Monnet, G. Sicard, F. Kastensmidt, M. Renaudin, and R. Reis, "Comparing transient-fault effects on synchronous and on asynchronous circuits," in *15th IEEE International On-Line Testing Symposium*, June 2009, pp. 29–34.

[11] S. Peng and R. Manohar, "Efficient failure detection in pipelined asynchronous circuits," in *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, 2005, pp. 484–493.

[12] K. T. Gardiner, A. Yakovlev, and A. Bystrov, "A C-element Latch Scheme with Increased Transient Fault Tolerance for Asynchronous Circuits," in *13th IEEE International On-Line Testing Symposium (IOLTS 2007)*, July 2007, pp. 223–230.

[13] W. J. Bainbridge and S. J. Salisbury, "Glitch Sensitivity and Defense of Quasi Delay-Insensitive Network-on-Chip Links," in *15th IEEE Symposium on Asynchronous Circuits and Systems*, May 2009, pp. 35–44.

[14] P. McGee, M. Agyekum, M. Mohamed, and S. Nowick, "A Level-Encoded Transition Signaling Protocol for High-Throughput Asynchronous Global Communication," in *14th IEEE International Symposium on Asynchronous Circuits and Systems*, 2008, pp. 116–127.

[15] F. Huemer, R. Najvirt, and A. Steininger, "Identification and confinement of fault sensitivity windows in qdi logic," in *2020 Austrochip Workshop on Microelectronics (Austrochip)*, Oct 2020, pp. 29–36.

[16] P. Behal, F. Huemer, R. Najvirt, A. Steininger, and Z. Tabassam, "Towards explaining the fault sensitivity of different qdi pipeline styles," in *2021 27th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2021, pp. 25–33.

[17] Z. Tabassam, P. Behal, R. Najvirt, and A. Steininger, "Input/output-interlocking for fault mitigation in qdi pipelines," in *2021 Austrochip Workshop on Microelectronics (Austrochip)*, 2021, pp. 17–20.

[18] J. Sparsø, *Introduction to Asynchronous Circuit Design.* DTU Compute, Technical University of Denmark, 2020.

[19] A. J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," in *Proceedings of the Sixth MIT Conference on Advanced Research in VLSI*, ser. AUSCRYPT '90. Cambridge, MA, USA: MIT Press, 1990, pp. 263–278. [Online]. Available: http://dl.acm.org/citation.cfm?id=101415.101434

[20] Z. Tabassam and A. Steininger, "Towards resilient qdi pipeline implementations," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022.

[21] si2.org. (2014) Silvaco open-cell 15nm library v0.1_2014_06 from si2. [Online]. Available: https://si2.org/open-cell-library/

[22] K. S. Stevens, D. Gebhardt, J. You, Y. Xu, V. Vij, S. Das, and K. Desai, "The future of formal methods and gals design," *Electronic Notes in Theoretical Computer Science*, vol. 245, pp. 115–134, 2009.