SET Effects on Quasi Delay Insensitive and Synchronous Circuits

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

***Abstract*—Due to their unbounded data accepting windows asynchronous circuits seem to be more susceptible to environmental effects than their synchronous counterparts with their strict data latching protocol. The technology advancement makes single event transients (SETs) more of a concern towards reliable operation.**

To better understand the properties of the mentioned classes we present their behaviour under the influence of SETs in a more detailed view that helps to visualize their unseen characteristics. For comparison we propose a way of fault injection where the length of a fault pulse is not fixed, calculated based on maximum gate delay, but related to the circuit's computation steps instead.

The analysis concludes that asynchronous quasi delay-insensitive (QDI) circuits show better resilience against SETs due to two main reasons: (1) if realized with a 4-phase handshake protocol they are 95 to 97% resilient to negative fault pulses (2) the susceptibility of a circuit is largely unchanged for increasing fault length because of the causality underlying the QDI principle.

Our analysis provides insights leading towards more resilient QDI circuits: if we only make a circuit or specific gates better resist "1" faults, we are fully resilient towards the single event transient (SET)s because "0" faults are already filtered out by its inherent behaviour. This is also beneficial for area efficiency; as asynchronous circuits often require already double or more area and computation time compared to synchronous circuits, adding extra SET mitigation with double-up or other buffer redundant techniques tends to result in painful overheads. Being able to focus the protection to "1" faults, as indicated by our analysis, can hence yield important savings.

I. Introduction

The benefits of asynchronous circuits have got into focus again in context with new architectures [1]–[4]. However, technology advancements pose some challenges, one being SET effects [5], [6].

Asynchronous circuits are known for their highly robust timing and superior energy efficiency [7] but the QDI variant seems a weak candidate towards SETs because it remains open to environmental effects due to its unbounded data accepting windows [8]. [9] already investigate QDI circuits in this respect in a detailed manner. Here we want to view their real behaviour in direct comparison to synchronous circuits under equal conditions. The answer shall clarify whether the strict latching protocol really protects the circuits from SETs on a long run.

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

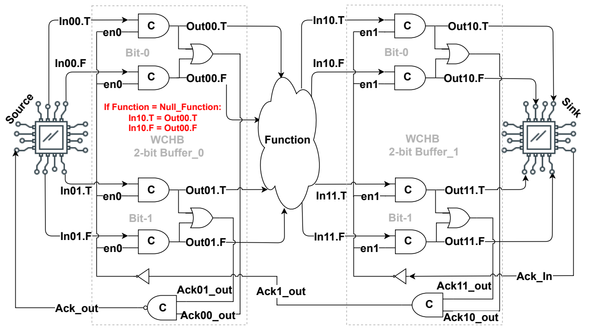

Fig. 1. 2_bit 2_stage WCHB

A. Asynchronous QDI logic

Instead of one control signal, namely the global clock in synchronous settings, asynchronous circuits are driven by local handshake signals. If a *source* initiates the handshake cycle: validity of data is indicated to the *sink* (or receiver) via a predefined protocol, and in response to this the sink generates an *acknowledgement* signal on an explicit line.

Within the basic styles of asynchronous logic the most robust one with respect to timing is the delay-insensitive (DI) one. In DI the data itself indicates the validity by using different encoding schemes. In this study we only consider dual-rail (DR) encoding, for each *bit* two rails are required, "bit.T" and "bit.F". A logical "1" is presented with "bit.T = 1, bit.F = 0" and a logical "0" with "bit.T = 0, bit.F = 1". These are the two valid data codes called *tokens*. The 4-phase handshake protocol is obeyed here, which means after each valid token we have to set both rails to low "bit.T = 0, bit.F = 0", which is called *spacer*. The 4-phase cycle comprises: the reception of a token, which is completed with a logic "1" on *acknowledgement*, followed by the spacer, and finally resetting *acknowledgment* again to low. If both rails go high simultaneously just like "bit.T = 1, bit.F = 1" this is considered illegal in this protocol.

When the DI circuits are realized with one extra constraint, namely *isochronic forks* (fork delays are matched), they are considered as QDI class.

B. Muller C-element (MCE): The elementary building block of QDI circuits

The MCE only responds to the inputs when they are matching, and it retains this matched pattern until the next matching pattern appears on its inputs line, which must be of opposite logic level to become stored. There are several symbolic representations, but in this article we use the one similar to ”AND_gate” with label ”C” on it as shown in Fig. 1.

C. *Weak-Conditioned Half Buffer (WCHB)*

Throughout our analysis of QDI circuits we consider WCHB [10] as our buffer template. We stick to the standard buffer style without any fault mitigation enhancement. Fig. 1 is a 2-bit, 2 stage pipeline circuit with WCHB as pipeline buffer/register. With the help of Fig. 2, ”FF” part, the functionality of the WCHB can be illustrated:

1) P1: The ”buffer_0” only allows a new token, if the ”buffer_1” contains a spacer, which means ”en0” must high.
2) As we are using DR encoding, only either the ”.T” or the ”.F” rail shows a transition in both, ”Bit-0” and ”Bit-1”.
3) P1*, P1**: Suppose ”In00.T” from ”Bit-0” and ”In01.T” from ”Bit-1” go high, which means both buffers will store a logical ”1”.
4) P2: Successful latching is indicated by toggling ”Ack_out” to high, which constitutes a flag for the predecessor.
5) Suppose, for simplicity, the logic function in between (”cloud”) is transparent. In that case the latched data simply passes to ”buffer_1”.
6) P3* and P3**: Source generates a spacer in response to P2.
7) P4: Whenever the spacer gets latched ”Ack_out” goes high again.
   The communication between ”Buffer_0” and Buffer_1 follows the same protocol.

The data-passing is straightforward as explained, where the important thing to notice is the generation of the ”Ack_out” signal that is only possible when all bits are completed, which is checked and retained by a MCE.

II. The Experiment Setup

We choose a pipelined multiplier circuit (multiplication is performed by a binary multiplication method) with our target circuits exhibiting different bit widths, including 16, 64 and 256. The multiplier is realized in three different variants, namely (1) in the synchronous style, and with QDI asynchronous design using (2) Delay-Insensitive Minterm Synthesis (DIMS) logic [11] and (3) NCLX logic [12]. In the latter two, the buffer style is the same, namely WCHB, while the only difference is in the realization of the logic.

A. *Robustness test environment*

Our simulation environment comprises of:

1) Data source and sink.
2) Device under test.
3) Monitors placed at the inputs and outputs of the device under test.
4) A database for storing results that deviate from a golden run.

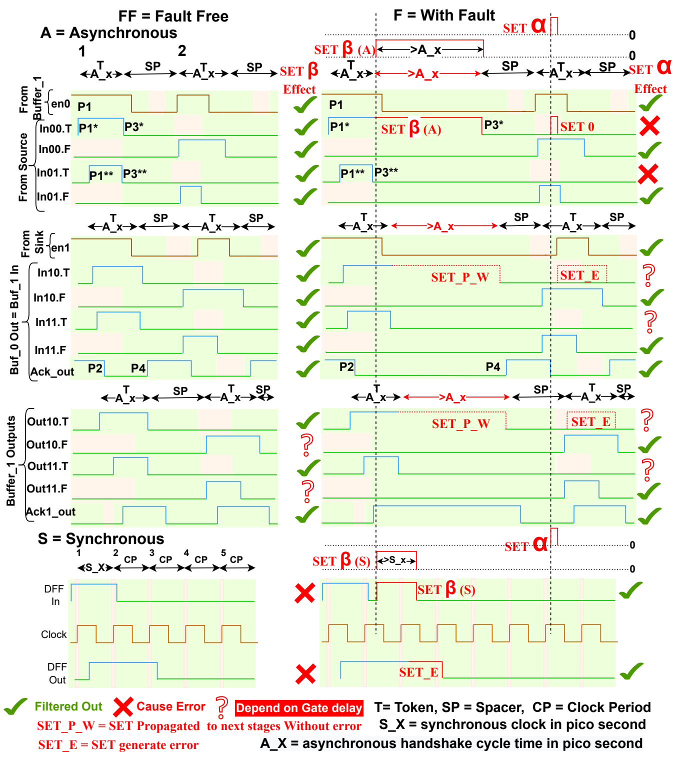

Fig. 2. Target circuits behaviour with and without fault scenarios

In this study we are investigating SET effects, so, to mimic these we inject one fault per simulation and run 3553534 simulations for three different types of circuits with three different data widths, while injecting 5 different lengths of fault pulses δ. The results are presented in Fig. 4. We are not injecting faults at inputs and outputs of the circuit, because in this setting the monitors directly observe these, so the result is trivial. Similarly, the clock line of the synchronous variants is excluded from injection. Time and location of injection are chosen randomly, where the number of faults is directly proportional to the number of nodes and the total simulation time of the respective circuit for predefined samples (δ’s).

The circuit gate delays are defined by using the NanGate 15nm open source library file with typical conditions from [13]. As asynchronous circuits utilize the MCE as storage element, that is, unfortunately, not part of this library, we use the NAND-gate model from [14] for its simplification to derive delay assumptions. An interpolation method is used for the calculation of gate delays, where we considered the same fixed value for index_1 that is $input\_net\_transition$ because this parameter is fixed in our simulations, while the $total\_output\_net\_capacitance$ represented by index_2 is based on the fan-out of the gate.

B. *Effective Pulse length δ*

[5] shows how the technology advancement affects the masking capabilities of circuits. As we are only interested in logical and temporal masking in our analysis, it is important

to inject a fault of length higher than the minimum gate delay in the circuit to visualize the circuit behavior, otherwise it will be filtered-out by the gate delays (electrical masking). In Fig. 3 we extract the gate delays from all of our target circuits and segregate into 5 categories listed from A% to E%. Fig. 3 concludes the lower bound for δ that is $50ps$.

In contrast, SETs in sequential circuits are only problematic when converted to single event upset (SEU)s, so only knowing the gate delays is not sufficient for the selection of an appropriate δ.

As shown in Fig. 2 part "FF", the asynchronous "buffer_0" only passes two data tokens, while the synchronous one passes 5 in the same amount of time (rough assumptions for understanding the fact). So if we select a δ based on the asynchronous requirement for observable results, it would be totally unfair to apply it for the synchronous target as well, because it will definitely show effects and not get temporally masked: The probability of hitting the sensitive window of a flip flop is equal to δ divided by the clock period [15] in synchronous circuits.

If we pick the statistics from the 4x4 asynchronous multipliers in Fig. 4, one handshake cycle period is around 1200ps, while the 4x4 synchronous multiplier clock period is 157ps. If we select 32% of these 1200ps as δ, that amounts to 384ps. With this selection, the comparison is totally biased because the probability to hit the sensitive window in the synchronous circuit is 100%. Therefore, for this work we choose the fault pulse lengths relative to the target circuit's global clock period for synchronous and worst handshake cycle time for asynchronous circuits.

As shown in Fig. 4, we compare the 4x4 asynchronous multiplier with the synchronous implementation while injecting 384ps (32% of 1200ps) and 50.24ps (32% of 157ps) respectively.

C. Error types

In synchronous circuits it is obvious that if we inject faults in all internal signals of a circuit except the clock signal, the only expected effect to observe is that the generated result is not correct (in the value domain). In contrast, in asynchronous targets there are more possible effects, as listed by [16]. To keep the focus and make it simple, in this article in the first phase we sum up all possible types into only one flag called *errors*.

III. Result Analysis

Figure. 4 reveals one very interesting property of QDI circuits towards the SETs; their resilience is not much affected by the fault length. This property can also be observed in the comparison results of [17] and [18]. In the synchronous counterpart the increasing trend of error rate with increasing δ is obvious, because the probability of hitting the sensitive window goes higher. The question arises why asynchronous circuits are insensitive to this, at the same time when we know from the inherent behavior of QDI discussed by [8] and from Fig. 2, that the sensitive windows (red shade) are

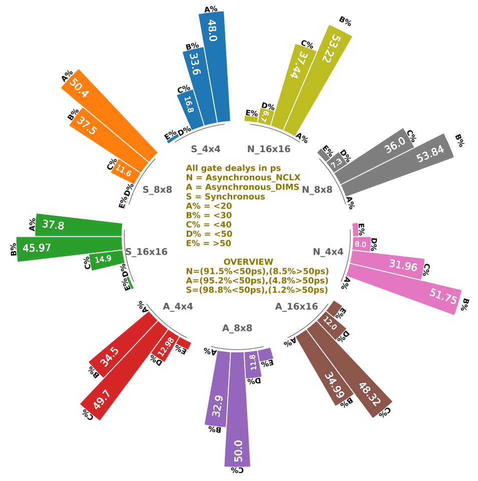

Fig. 3. Overview of target circuits gate delays

much bigger and not precisely predictable due to the delay insensitive communication protocol. For better understanding, we will investigate this in more detail in following sections.

A. Error Contributors

Fig. 5 is comprised of two parts; first we discuss the "SETs at the buffer's input/outputs or in logic" (where the results are extracted from all target variants). Before starting the discussion let us recall the fact that SETs are only problematic when converted to SEUs, .i.e., when the pulse on the signal actually influences the *state* of the circuit. This can be due to two effects:

1) When a fault hits within the logic and propagates to a storage element input, thus being converted into an SEU.
2) When a fault directly hits a storage element's inputs or outputs and gets retained.

Accordingly, in the figure the first case is indicated by blue color of the bar, while the latter case is shown in yellow color, with mixed cases shown in a continuous change of color.

In the synchronous case the causes are mixed, while for QDI with DIMS logic represented by "A_" the situation is very different in that the main contributor is logic. This is because DIMS logic [11] contains numerous MCEs that are basically storage elements as well. So, in this case the computation logic is not only propagating the errors (like in the synchronous case) but also converting these into SEUs.

From Fig. 4 it is visible that "N" is most resilient towards SETs. NCLX logic still utilizes MCEs for completion detection in logic, but only simple gates for computational logic, consequently the contribution of logic to effects is not as much as in DIMS. The trends of "N_" in Fig. 5 are very insightful in that they also back the argument presented for DIMS logic: As we know for the 4x4 multiplier the number of bits is lower than for 16x16, so moving from "N_4x4" to "N_16x16" the error contribution changes from buffers towards logic. As in the 16x16 NCLX multiplier the portion of completion detection in logic increases because of the higher number of bits, we

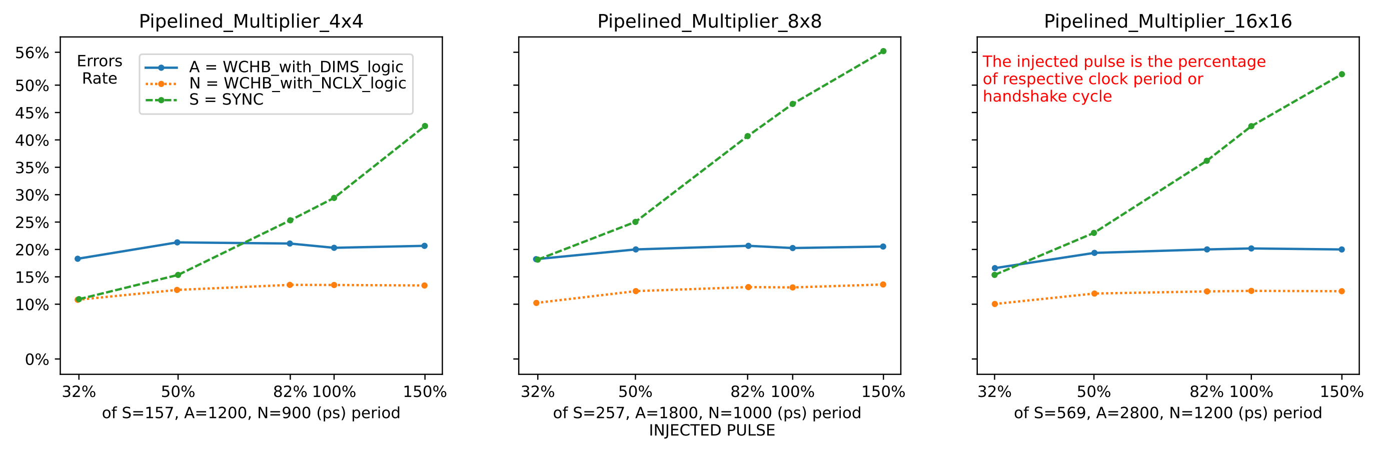

Fig. 4. Error rates over fault length as observed for the different targets and design styles in our experiments

have a higher number of MCEs compared to the 4x4 multiplier that contributes more.

The second part of Fig. 5 gives insight about how much the circuit is sensitive to fault polarity, i.e. to 1- and 0-pulses. Unsurprisingly, synchronous circuits show the same susceptibility towards 0 and 1. However, asynchronous circuits show 95 to 97% resilience towards 0. In this sense asynchronous circuits show more robust behavior compared to synchronous ones towards SETs. But the question arises why these are not influenced by ”0” SETs.

B. *QDI Circuits are largely insensitive to ”0” Faults*

The main reasoning is given here and illustrated using Fig. 2 ”FF”. Here we restrict our explanation to ”In00.T” and ”In00.F”, that is ”Bit_0” of ”Buffer_0” from Fig. 1. The behavior of the other signals is analogous.

1) In the asynchronous circuit during window ”T” only one input shows a transition to logic 1 (dual-rail coding), so the other input, in this case ”In00.F”, remains at logic 0. As a consequence, the latter is fully resilient to a 0 fault. Alternatively, if the 0 fault hits ”In00.T” it only causes a timing issue: The logic 1 will vanish earlier or arrive later, and due to the protocol’s delay insensitivity normal operation continues after the transient without effecting the result’s integrity.
2) If the 0 fault effects the ”en0” signal during ”T”, it also causes a timing issue. We know ”en0” is sourced by the completion detection signal that is again generated from more than one buffer and stored in the MCE, so it is hard for an SET to permanently corrupt a signal that is backed by multiple sources. As presented in Fig. 1, the ”Ack_out” is generated using ”Ack00_out” and ”Ack01_out” and retained using the MCE.
3) During ”SP” we know that all, ”en0”, ”In00.T” and ”In00.F”, are already on low level so ”0” faults are harmless in this region. If one appears, it can at most cause a timing issue that is totally fine in QDI logic.

C. *QDI Circuits are Insensitive to Fault Pulse Length δ*

As already discussed the increase in δ will increase the probability of hitting the sensitive window in synchronous circuits which is clearly evidenced by the results presented in Fig. 4. On the other hand the results for our asynchronous targets show that their sensitivity is not directly proportional to the length of faults. This is because their sensitive windows are defined by the input/output signals (environment), where the latching of data is only possible if they meet the protocol conditions. Therefore, the sensitivity depends on the time of fault relative to the handshake timing, and not on the length.

If the fault pulse appears and is not latched in the same cycle in a synchronous circuit, the probability remains that it is long enough for hitting the latching window in the next cycle, as presented in Fig. 2-”SET β (S)” part ”F”, and not logically masked otherwise. However, in the asynchronous circuit the causal behavior ensures that if the pulse is logically masked in the cycle where it occurred, it must have vanished before the next cycle can start in normal situations.

1) To make things simpler we only inject on and illustrate the behavior of ”In00.T” under fault, while the results for other rails are concluded with the symbols ”Filtered Out”, ”Cause Error” or ”Depend on Gate Delay” in Fig. 2 ”F”.
2) ”SET β (A)” appears at ”In00.T”. This rail is already set to high by the source and about to go low (from ”FF”-P3*), since ”Ack_out” is already activated, as indicated with P2.
3) Due to the SET ”In00.T” remains high for the duration of the fault that is $>$ ”A_x”.
4) As a result, this SET only keeps the circuit from latching the spacer for some time; after that the circuit continues its operation without any issue, regardless of the fact that the SET propagated to the next stage as highlighted with ”SET_P_W”.
5) As shown ”Ack_out” is generated after the SET vanished at P4.

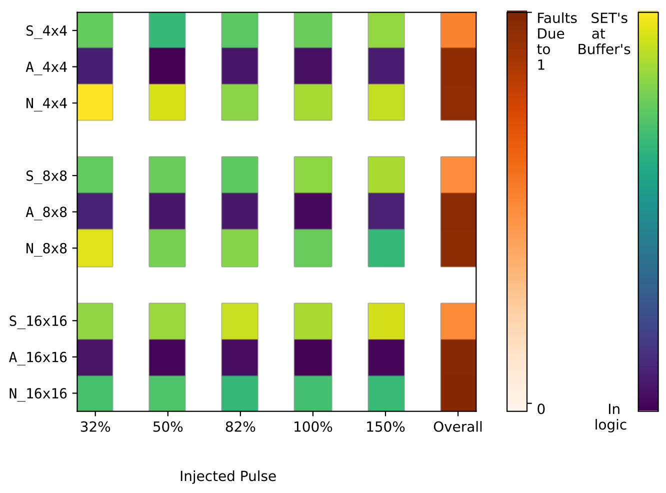

Fig. 5. Error Contributors

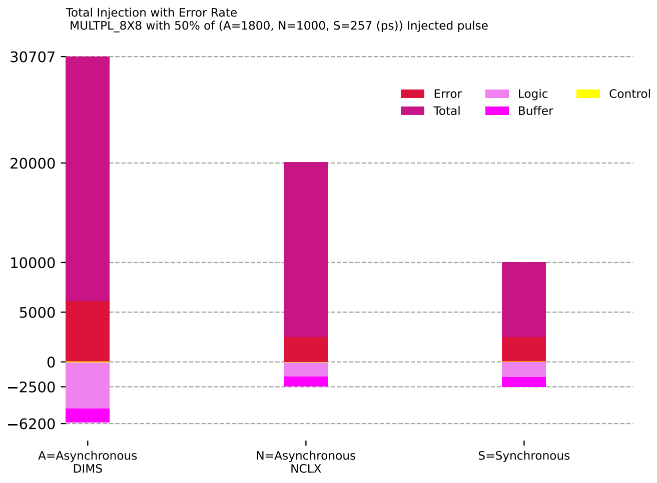

Fig. 6. Error/Injection Details

6) The important thing is that "SET β (A)" is not affecting any rail of the asynchronous circuit, as shown with the symbol "Filtered Out", because it appears in the green window and only causes delays in the handshake cycle. The only uncertain situations are for "Out10.F" and "Out11.F", because the SET appear very close in time when the sink responds, and if it delays a bit, the SET is considered as valid.
7) In contrast to SET β the "SET α" in Fig. 2 is very small compared to the handshake cycle "A_x", but it appears when most of the rails are sensitive to "1" fault. As shown it appears during the red window of "In00.T", and as its length is greater than the respective gate delay, it is latched and propagated into the pipeline represented by "SET_E".
8) Rails marked with symbol "Cause Error" or "Depend on gate delay" are susceptible if the fault hits one of these.
9) Notice how in this case the synchronous circuit simply does not react to "SET α", because it appears outside the red window.

This analysis concludes that the susceptibility of an asynchronous (QDI) circuit depends on the time of fault occurrence relative to the handshake process, not on fault length. In contrast, synchronous circuits start losing robustness when the SET length goes higher than their clock period. This is an important insight, especially considering that we have used *relative* pulse length: As features get smaller in VLSI technologies, SET effects tend to get more massive, hence also causing longer pulses, especially when related to the clock periods that tend to decrease over technology [5]. Asynchronous targets, in contrast, do not seem to significantly suffer from this trend – for the reasons explained above.

D. Error/Injection Overview

Figure. 6 gives more insights about the results of our fault injection experiments. The data shown are extracted from an 8x8 multiplier whit δ equal to 50% of the respective period. The number of total injections in the asynchronous "DIMS" is approximately twice as many as for synchronous. This is because our strategy was to keep the number of injections proportional to the area of the target circuit and its computation time.

On the positive axes the red shade shows the error rate relative to the number of total injections. These rates are more elaborated on the negative axes to give insight about the main contributors. In "asynchronous_DIMS" the logic is the main contributor, which is compensated if we realized these with "NCLX" logic. It is important to note that control signals are not contributing much. This is because the control signals are more protected, as discussed in Section. III-B-2. Recall that for synchronous circuits there are no contributions from a control signal, as we are not injecting into the clock.

E. Robustness of QDI circuits with unbalanced timing

Recall from Fig. 2 that the timing of QDI circuits is defined by inputs (from "Source") and "acknowledgments" (from "Sink"). So far in our analysis related to Fig.4 we only considered the results for SETs when the circuit is in balanced mode. Apart from the situations when the SET causes some extra delays that are not in our control, we did not consider scenarios where any side of the environment behaves asymmetric (adds extra waiting time). This clearly needs to be completed to mimic the real behavior of asynchronous circuits.

So, in this section we analyze the behavior of QDI circuits against SETs with unbalanced circuit dynamics.

The variable source and sink delays are sometimes collectively subsumed in a Pipeline load factor (PLF), where [16] presents the behavior of QDI circuits against SETs with variable PLF. In short if "Source" is slow we say the circuit is in "Token limited mode" where "PLF $<$ 1", while in the opposite direction when "Sink" is slow the circuit is in "Bubble limited mode" with "PLF $>$ 1".

To verify whether our analysis also holds with unbalanced behavior of QDI circuits, we performed studies for a PLF

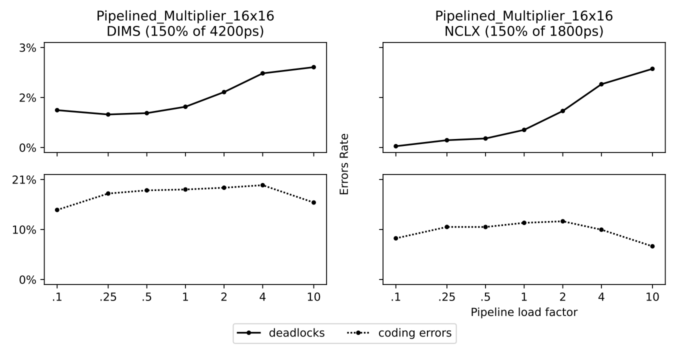

Fig. 7. Error rate with different PLF's

different from 1. We only present the results of the 256 bit asynchronous multiplier with "DIMS" and "NCLX" logic while injecting 150% of their respective periods in Fig. 7. In Fig. 4 we combine "Deadlocks" and "Coding errors" in a single flag "Error rate" while in Fig. 7 we presents these separately. Fig. 7 concludes that the circuit resistance remains roughly the same for varying PLF, the only differences show up for extreme cases of PLF. Towards these corners the coding errors go lower, at the cost of deadlocks. This is not unexpected because the circuit response is more delayed for extreme PLF, so the monitor simply flagged a deadlock, if, due to an internal circuit error, valid results are not produced within some predefined time-out.

The experimental results of WCHB with "DIMS" presented by [16] is not directly comparable with Fig. 7 because they run fault injection simulation with different δ and different settings, but the trends are same. Coding errors approximately remain the same for PLF > 1 to 4 and show a decreasing trend for PLF < 1. But here our main concern is to check whether the QDI circuit drastically changes its behavior with unbalanced pipeline timing – which is not the case as shown in Fig. 7.

IV. Conclusion

We have performed extensive fault injection experiments into target circuits implemented in the synchronous design style as well as two asynchronous QDI styles, namely DIMS and NCLX, both with 4-phase protocol and dual-rail data encoding. The aim was to perform an apples-to-apples comparison of the robustness of the respective circuits towards SETs. Our setup is specific in choosing the fault length relative to the clock/handshake cycle time of the respective implementation, thus naturally considering different operational speed.

Key conclusions of our analysis were that asynchronous circuits, beyond their known robustness against delay variations, exhibit a high degree of resilience against negative fault pulses, due to the return-to-zero property of the 4-phase protocol. In addition, their transition-based operation gives them an edge towards fault lengths that are even greater than their cycle time. Rather than causing data corruption, many of the long fault pulses are naturally converted into extra delays, which can be easily managed by the delay-insensitive circuit design.

Synchronous circuits, in contrast, are quite resilient for fault lengths covering less than a clock cycle. With technology advancement [5], however, SETs tend to become longer relative to the clock period, hence increasingly challenging their robustness.

Our insight that positive fault pulses are the main contributors to problematic SET effects in asynchronous circuits provides important guidance for our future work that will be concerned with SET assessments and mitigation techniques.

References

[1] D. Bertozzi, K. Bhardwaj, and S. M. Nowick, "An asynchronous soft macro for ultra-low power communication in neuromorphic computing," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022, pp. 178–181.

[2] Z. Li, Y. Huang, L. Tian, R. Zhu, S. Xiao, and Z. Yu, "A low-power asynchronous risc-v processor with propagated timing constraints method," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 9, pp. 3153–3157, 2021.

[3] S. Nelson, S. Y. Kim, J. Di, Z. Zhou, Z. Yuan, and G. Sun, "Reconfigurable asic implementation of asynchronous recurrent neural networks," in *2021 27th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2021, pp. 48–54.

[4] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[5] D. Kobayashi, "Scaling trends of digital single-event effects: A survey of seu and set parameters and comparison with transistor performance," *IEEE Transactions on Nuclear Science*, vol. 68, no. 2, pp. 124–148, 2020.

[6] P. Tsoumanis, G. I. Paliaroutis, N. Evmorfopoulos, and G. Stamoulis, "Analysis of the impact of electrical and timing masking on soft error rate estimation in vlsi circuits," *Technologies*, vol. 10, no. 1, p. 23, 2022.

[7] N. L. V. Calazans, T. A. Rodolfo, and M. L. Sartori, "Robust and energy-efficient hardware: The case for asynchronous design," *Journal of Integrated Circuits and Systems*, vol. 16, no. 2, pp. 1–11, 2021.

[8] 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.

[9] A. A. Sakib, "Soft error tolerant quasi-delay insensitive asynchronous circuits: Advancements and challenges," in *2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2021, pp. 1–6.

[10] P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A designer's guide to asynchronous VLSI*. Cambridge University Press, 2010.

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

[12] A. Kondratyev and K. Lwin, "Design of Asynchronous Circuits by Synchronous CAD Tools," in *Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324)*, June 2002, pp. 411–414.

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

[14] 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.

[15] E. Costenaro, D. Alexandrescu, K. Belhaddad, and M. Nicolaidis, "A practical approach to single event transient analysis for highly complex design," *Journal of Electronic Testing*, vol. 29, no. 3, pp. 301–315, 2013.

[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] B. Rahbaran and A. Steininger, "Is asynchronous logic more robust than synchronous logic?" *IEEE Transactions on dependable and secure computing*, vol. 6, no. 4, pp. 282–294, 2008.

[18] 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.