Input/Output-Interlocking for Fault Mitigation in QDI Pipelines

Zaheer Tabassam, Patrick Behal, Robert Najvirt, and Andreas Steininger
Institute for Computer Engineering, TU Wien, Vienna, Austria
{ztabassam,pbehal,rnajvirt,steininger}@ecs.tuwien.ac.at

***Abstract*—In asynchronous quasi delay-insensitive (QDI) circuits, temporal masking is a serious concern because of their event-driven behavior, which makes them prone to environmental effects: Data acceptance windows, e.g. are defined by transitions (token/acknowledgement) alone, without temporal bounds, therefore a glitch occurring anytime throughout such a window cannot be distinguished from an expected, correct transition in a straightforward manner and hence threatens data integrity. Therefore, shortening that window is one proposed way in the literature to enhance temporal masking in QDI designs.**

We examine a variant of the Weak-Conditioned Half Buffer (WCHB) called Interlocking WCHB (which wisely shortens the transition window) because of its glitch filtering properties and a low cost implementation as compared to other variants. We propose modifications that enhance its dealing with illegal token words specifically when waiting for acknowledgment signal transitions in the so-called bubble limited operation mode. A very strict triple-check input filter with a glitch filter preventing the buffer from capturing an illegal state is used, which also enhances the deadlocking rate of the circuitry.

I. Introduction

Due to the worsening delay variations caused by the ever shrinking technology node sizes, industry is starting to shift from synchronous designs, where the worst case timing of the critical path limits the maximum clock speed, to asynchronous designs where the throughput is mostly limited by the average computation time. Additionally, the transient fault rate is also increasing, and due to the less rigorous timing assumptions of asynchronous QDI circuits compared to synchronous circuits it is harder to mitigate them.

In the literature, there are many different approaches to increase the resilience of asynchronous circuits with respect to transient faults: comparison with duplication, triple modular redundancy (TMR) and shortening of fault sensitivity windows to name some. After thorough analyses of existing mechanisms, we chose the Interlocking WCHB for its good filtering properties as well as its very low area overhead.

We propose improvements to the Interlocking WCHB in this paper for circuits in bubble-limited operation mode, where faulty and valid data at the dual-rail (DR) inputs are latched at the same time on the arrival of the acknowledgment signal, which can generate an illegal token. As the original interlocking mechanism only works after the first transition to high on the DR output, an input filter is introduced to check the validity without the dependency on the acknowledgment signal. Another noticeable effect is that the modification also acts like a small glitch filter and therefore short pulses are hindered from reaching the input of the latch at all.

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

II. Related work

The main focus of our research are transient faults in QDI circuits, where we survey models, their effects and hardening techniques for dealing with them. Synchronous systems are inherently quite fault tolerant due to strict clock dependent transitions and there are a lot of established techniques for hardening, if required. Unfortunately, these techniques are not easily applicable for asynchronous circuits and show large overheads and constraints [1]. In [2], the sensitivity to transient faults of asynchronous logic blocks is compared to synchronous ones and the respective masking effects are analyzed. In [3], different methods are modified for the asynchronous domain. In general, some form of redundancy is always required for error detection as in [4] and [5].

A prominent technique [6] is to cross check the results of double up circuits to ensure SET tolerance. Based on this method a hardened QDI processor has been presented in [7].

In [8], [9], [10] and [11] different types of redundancy methods are introduced to also address fault issues. Also focusing on the reduction of sensitive windows, more than 10 single event transient (SET) mitigation techniques are presented in [12].

Using the WCHB as a reference and few notable implementations from [12], [13] and [6], authors in [14] perform fault injection simulations with a special visualization method to identify the fault sensitivity windows in several QDI logic styles and compare their tolerance with two proposed half-buffer designs.

III. Asynchronous logic

Asynchronous circuits are similar to synchronous ones with respect to the data path, but the main difference is the use of local handshakes for timing control instead of a global clock. In this handshake process the sender ("source") indicates the validity of a new data word ("token") by activating a request (req), while the receiver ("sink") confirms its reception by an acknowledgement (ack).

The delay-insensitive (DI) class of asynchronous circuits is the most flexible in timing, as data itself defines its own validity (through DR encoding for each data bit) thus obviating the need for a dedicated req signal along with its timing assumptions. With their extra assumption of matching delays at

some forks, QDI circuits represent a more practically realized class of DI.

In the DR encoding a signal x is represented by two rails $(x.0, x.1)$, the data value $x = HI$ being represented by $(0, 1)$, and $x = LO$ by $(1, 0)$.

The commonly used 4-phase handshake protocol demands a null phase $(0, 0)$ as a separation between any two such data values. The pattern $(1, 1)$ is unused and it represents an illegal code word.

A. Fault Sensitivity Windows

The basic building block of asynchronous circuits is the Muller C-element (MCE), which changes its output to the logic value of its inputs if all inputs are equal. For the commonly used 2-input MCE, this means that, starting from two matching inputs, as soon as one input change arrived, the element waits for the second input to transition as well, and during this waiting time (sensitive window) it is prone to any faulty transition which will erroneously flip the output. Compare this to the synchronous register which samples its input only at one instant, namely the active clock edge. In [14] a comprehensive view of these fault sensitivity windows is presented. The authors also illustrate the behavior with variable source and sink delays, to have the circuit operate in token and bubble limited operation. In the former, the buffer stages are mostly waiting for new data to arrive (they are empty) while in the latter, they are waiting for the acknowledgement (holding a valid data word). To measure the load of the circuit we use the Pipeline load factor (PLF). A high PLF indicates that the target operates in a bubble limited mode and, on the other hand, a small PLF (< 1) indicates token limited operation.

IV. Baseline Buffer Template

Asynchronous circuits are often split in small portions which are pipelined to leverage concurrency for higher throughput. Like the flip flops in synchronous pipelines, buffers serve as temporary data storage in asynchronous pipelines. From the numerous existing 4-phase QDI buffer templates, we considered the simplest one, namely the WCHB as our baseline. In [14], its fault sensitivity is assessed through a visualization of its sensitive windows, i.e. the time intervals when transient faults hitting a certain signal have an effect on the output. The following effects are considered:

- Deadlock: The circuit stalls
- Glitch: A handshaking protocol violation is observed
- Value errors: The result value is wrong
- Code errors: An illegal code is observed
- Timing deviation: Timing is changed

These error types are discussed in [14] in more detail.

To reduce the sensitive window, the authors in [14] propose circuit modifications like the Interlocking WCHB and Deadlocking WCHB. In this publication, for further investigation, the Interlocking WCHB variant of the WCHB serves as a reference template.

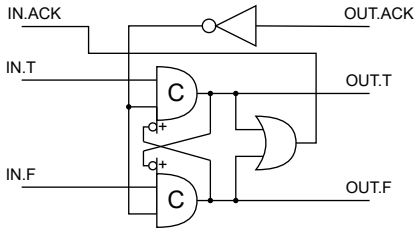

Fig. 1. Interlocking WCHB: Relative to the basic WCHB the cross coupling from one MCE output to the other MCE's "+"-input is added

A. Interlocking WCHB

The Interlocking WCHB [14] is designed to prevent the invalid $(1, 1)$ DR code word to be stored in the buffer with very little area and delay overhead. Its design principle is to accept the first high transition on a pair of MCE outputs that represent a DR bit, and to inhibit the same on the second MCE through interlocking, as presented in Fig. 1. As a consequence, only the first arriving transition is latched – which could be the faulty one, but the illegal code word is not able to propagate. Note however, that the interlocking does not work instantly; there is some propagation delay involved in locking the input of one MCE through the transition of the other.

In token limited operation, the ack transition arrives before the valid token, so if any rail shows a transition first, it is latched by the MCE and considered as valid. Therefore, the circuit either masks the fault (if the correct transition arrives first), or produces a value error, if the fault hits the rail that should have remained low before the intended transition arrives on the opposite rail. However, in the bubble limited case, the ack transition lags behind the data tokens, and therefore, both MCEs may try to fire at the same time (if both, the valid and a faulty transition arrived before the ack), and then the interlocking mechanism fails due to the mentioned delay the interlocking needs to work. This not only generates an illegal code word at the output of the buffer but also allows the latched code to propagate further through the pipeline due to the same reason. Therefore coding errors become more frequent with higher pipeline load factor as visualized in Fig. 2 (with a pipelined ALU circuit (ALUPL) as the target).

V. Proposed Modifications

The novel approach presented in this paper is the *InOutInterlock WCHB* shown in Fig. 3 as an enhanced version of

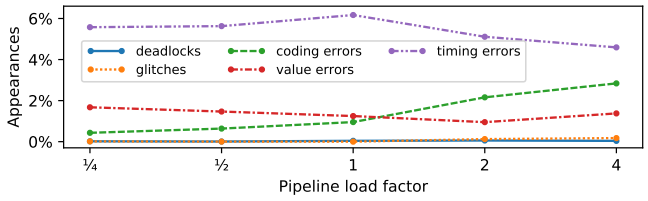

Fig. 2. Simulation results of ALUPL-4bit-DIMS using the Interlocking WCHB buffer.

the Interlocking WCHB to prevent the discussed coding error problem. The following modifications are applied:

- Asymmetric MCEs with one positive and one normal input are added for a first validity check. A rising transition on one rail on the normal input is only captured if the other rail is low.
- NAND gates serving as glitch filters are added. The input MCEs of the first check serving as delay elements, short positive pulses having a lower pulse width than the propagation delay of the MCEs are filtered from reaching the output MCE.
- As the NAND gate inverts the logic, there is no need for the acknowledgement signal inversion anymore. This saves some transistors and therefore contributes to compensate the area overhead of the additional input filter.
- Revision of the output MCE from two normal and one positive inputs used in the Interlocking WCHB to a two normal and one negative input to further reduce area overhead.

A. Working Mechanism

The first high transition on any input rail disables the input MCE of the other rail for a high transition similarly to the interlocking done at the output. The advantage is that this interlocking is done without any delay and therefore more effectively filters illegal code words. This input filter passes the token to the NAND gate glitch filter, which ensures that any pulse at the input shorter than the propagation delay of the input MCE is not propagated. Until the token is latched by the output MCE, the direct input of the NAND gate provides the possibility to flush a faulty transition latched by the input MCE such that short pulses on data rails have no effect other than a possible timing deviation, as they might temporarily block the correct transition to propagate. Output interlocking minimizes the probability of coding errors that may be generated due to faults hitting a NAND output or faults with a pulse width longer than the filtering threshold. As can be seen in Fig. 2, the probability of coding errors in the Interlocking WCHB grows with an increasing PLF. So, the proposed modifications (InOutInterlock WCHB) are most effective in bubble-limited mode (high PLF), as only the first transition is passed to the output buffer and will then be latched when the acknowledge signal arrives. In Fig. 4, the experimental results with the same target and configuration show that the InOutInterlock WCHB has better fault tolerance in most respects.

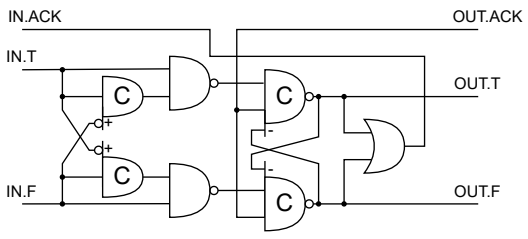

Fig. 3. InOutInterlock WCHB

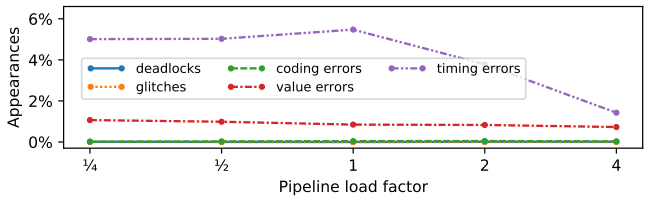

Fig. 4. Simulation results of ALUPL-4bit-DIMS using the InOutInterlock WCHB buffer.

VI. Results and Discussion

To validate the effectiveness of the new buffer style, we conducted a total of over 200 million gate level simulations on 5 different target circuits with a total of over 300 variations. In each simulation, one fault was injected by forcing a randomly selected signal to a random value at a random time for $1\,ns$. The following target circuits were used: a basic adder (ADDERPL), an ALU (ALUPL) a pipelined multiplier (MULTPL), an IIR filter (IIR), and an iterative multiplier (MULTIR). The following variations for each target were applied:

- Data-width: 4 and 8 bits (the results will be averaged)
- Logic-style: Delay-Insensitive Minterm Synthesis (DIMS) and NCLX [15]
- Buffer-style: WCHB, Interlocking WCHB and InOutInterlock WCHB
- PLF: 1/4, 1/2, 1, 2 and 4

Fig. 5 presents a comprehensive comparison between the baseline Interlocking WCHB and InOutInterlock WCHB with WCHB as the main reference. For the iterative targets (where a feedback in the data flow is used), it is not really possible to control the PLF with external delays and therefore we just show the results with bar graphs.

From the results the following observations can be made:

- In nearly all cases the InOutInterlock WCHB performs better than the Interlocking WCHB and the WCHB, the only exceptions being the MULTPL-NCLX for PLF of 1 where the deadlock performance is lower and ALUPL-NCLX for PLF below 1 where the value error rate is higher.
- In general for all experiments we see coding error rates very close or equal to 0% especially for higher PLF we can see potent improvement. This shows that the InOutInterlock WCHB does work for a wide variety of targets and conditions.
- Also for the value error category we can see a significant improvement. This can be explained by the glitch filtering capabilities of the InOutInterlock WCHB.

The error rates are only one part of the picture. We also need to investigate the introduced overhead of the new buffer style. In Fig. 6 the area overhead and the throughput reduction are shown compared to the WCHB for the DIMS logic style.

We can see that the Interlocking WCHB has a negligible area overhead compared to the WCHB, while the InOutInterlock WCHB performs much worse in this category. For

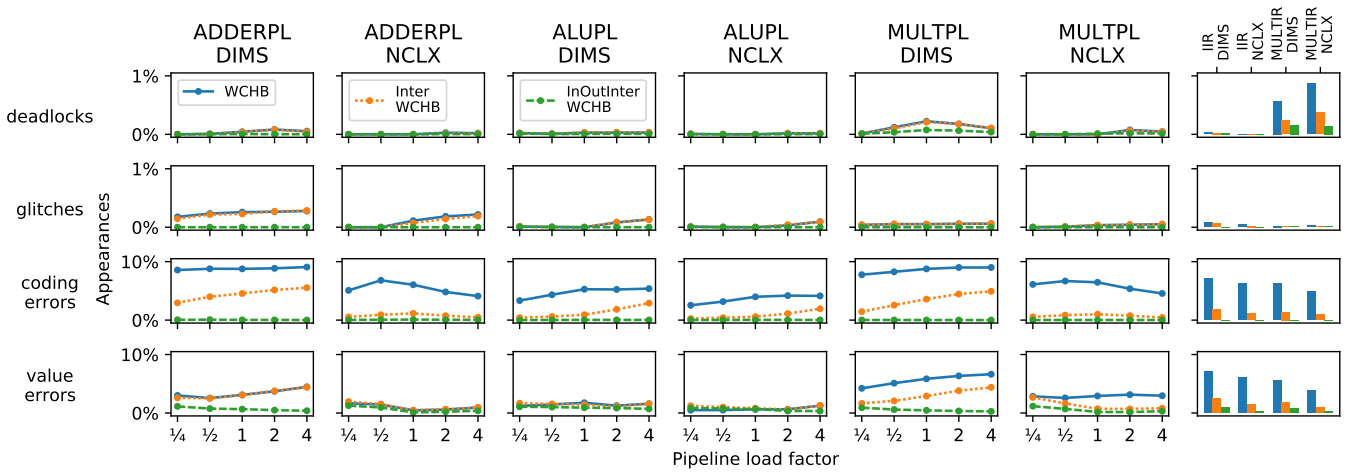

Fig. 5. Comprehensive Comparison between WCHB, Interlocking and InOutInterlock WCHB

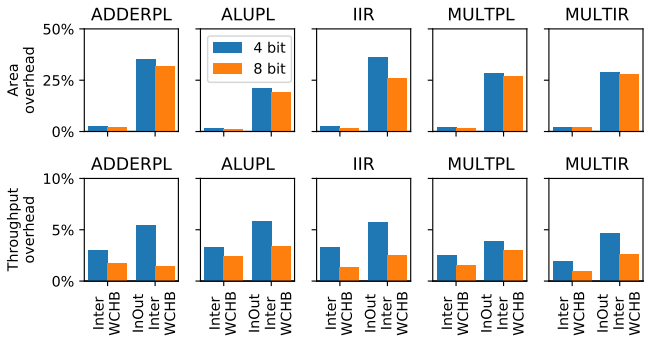

Fig. 6. Area and Throughput overhead of Interlocking and InOutInterlock WCHB

some circuits, where the improvement of fault tolerance over the Interlocking WCHB is less significant, the InOutInterlock WCHB might be considered too costly, but in general, an overhead of around 25% seems fair and affordable. Also for the throughput penalty over the WCHB, the InOutInterlock WCHB is nearly twice as bad as the Interlocking WCHB, but an absolute 5% increment is still relatively low.

VII. Conclusion

In this paper, we presented the InOutInterlock WCHB buffer which adds an interlocked input filter coupled with a glitch filter to the Interlocking WCHB buffer, correcting the most common reasons for its interlocking to fail. We could show that the additional filters significantly improved the SET fault tolerance for a wide variety of circuits and operating conditions. We also discussed the impact on area and throughput the improved buffer has.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[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] 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.

[13] 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.

[14] 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.

[15] 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.