On the Effects of Permanent Faults in QDI Circuits - A Quantitative Perspective

Raghda El Shehaby and Andreas Steininger
TU Wien, Institute of Computer Engineering, Vienna, Austria
relshehaby@ecs.tuwien.ac.at, steininger@ecs.tuwien.ac.at

***Abstract*—With their event-driven nature, quasi-delay-insensitive (QDI) asynchronous circuits offer a compelling fail-stop behavior along with an inherent 100% permanent fault detection coverage. In fact, permanent faults break the handshake, pushing the circuit into deadlock. In this paper we give quantitative results for the relative occurrence of the different effects that can manifest in a QDI circuit due to permanent faults. We study in which and how many cases the state of the circuit gets corrupted before reaching the deadlock. This behavior diagnosis enables us to identify the cases in which the circuit can go back to operating normally if a (self-) repair process were to take place. This investigation is conducted through extensive fault injection experiments in a chosen circuit simulation. Stuck-at faults are injected on a gate-level VHDL model of the circuit, with a wide coverage of parameters.**

***Index Terms*—permanent faults, QDI circuits, fault injection**

I. Introduction

Synchronous designs follow a rigid time grid dictated by a global periodic clock signal, where data validity is considered at specific clock ticks. Even though asynchronous circuits are more difficult to design than their synchronous counterparts, they have a lot to offer because their functional insensitivity to layout and parametric variations makes them correct by design. Quasi-delay-insensitive (QDI) circuits exhibit a natural resilience against delay variations, making them immune to delay faults [1], while still being vulnerable against value faults, which can either be transient or permanent.

In asynchronous circuits operation is driven by a causal chain of events, i.e., a transition on one signal causes a next one, and so on. The indication principle in addition requires each transition issued by a sender to be acknowledged by the receiver – a *handshake* typically implemented by some kind of request/acknowledge protocol – which yields a feedback loop that enables a self-oscillating behavior. Due to this property asynchronous circuits exhibit an inherent *fail-stop* behavior whenever physical damage causes a change in the structure of the circuit. A single permanent fault can break the handshake protocol and cause deadlocks [2]. The circuit stops operating in the absence of transitions needed to trigger further activity, and simply waits until the expected transition finally occurs, even if that should take very long and involve repair actions. In this ideal picture recovery is not needed; as soon as the repair is finished, the circuit continues operating properly.

This research was partially supported by the project ENROL (grant I 3485-N31) of the Austrian Science Fund (FWF) as well as the Doctoral College on Resilient Embedded Systems (DC-RES)

Unfortunately, there are cases in which the circuit's state does get polluted or incorrect outputs are produced as a consequence of the defect. Such cases have already been reported in the literature, but their relative occurrence rate has, to the best of our knowledge, not yet been fully explored, at least quantitatively. In this paper we will explore the possible behaviors of a QDI circuit under permanent defects, and we will associate probabilities with each of these.

II. Related Work

While most research focuses on transient faults, relatively little light is shed on the effects of permanent faults. In [3], the authors show that a stuck-at fault in a delay-insensitive (DI) circuit could either be *inhibiting*, preventing a transition from taking place, or *stimulating*, causing a transition to occur prematurely, or both. The analysis of the effects of stuck-at faults in [4] leads to the conclusion that DI circuits are always self-checking under the stuck-at fault model, while a QDI model can not be totally self-checking [5]. The latter work identifies possible fault effects in a QDI circuit as late detection, invalid transitions and premature completion. In [1], the authors additionally classify errors on a high level in terms of deadlock, synchronization failure, token generation and token consumption in QDI circuits.

Apart from some initial results in [6], we could not identify any *quantitative* statements on the behavior of QDI circuits under faults. In case of permanent faults, it has been identified in [1], [5], [7] that some type of *fail-safe* deadlocking behavior is preferable over a continued, erroneous circuit operation. However, it has remained unclear so far, with which probability an (unprotected) QDI circuit exhibits such behavior.

III. Permanent Fault Effects in QDI Circuits

QDI circuits operate correctly with positive, but unbounded, delays in gates and wires, while only imposing the *isochronic fork* restriction: a signal must arrive at all ends of the fork at the same time [8]. A QDI circuit will always eventually deadlock in reaction to a permanent fault [1], [9]. This is due to the indication principle which forces it to stop unless it receives an acknowledge for its previous signal transition.

A. User's Perspective

Based on this precondition, we classify the possible behaviors from a user's perspective, regardless of the circuit's internal operation.

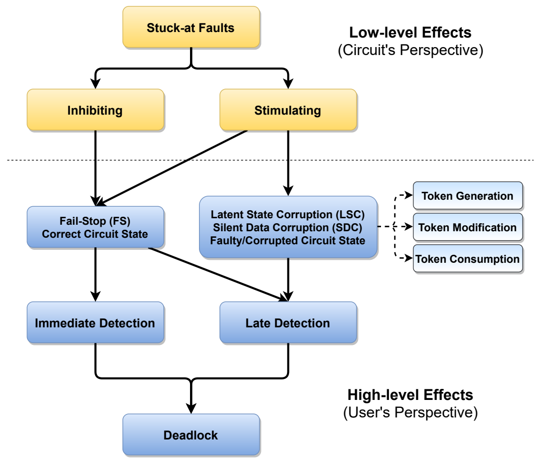

Fig. 1. Stuck-at Fault Effects Classification

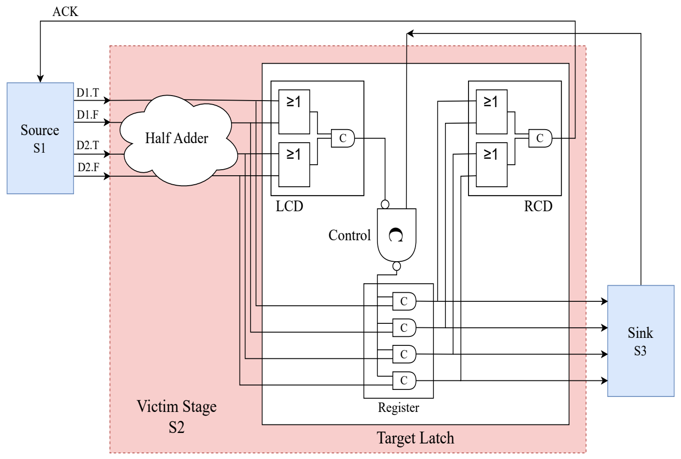

Fig. 2. Target Circuit

- *Silent Data Corruption (SDC):* The output latch captures incorrect data before the operation stops.
- *Fail-stop behavior (FS):* The circuit deadlocks without deviating from its fault-free operation until then. Recovery is not required: after a repair of the physical defect, the circuit simply continues with its operation correctly. This is an attractive property for self-repair.
- *Latent State Corruption (LSC):* The circuit stops operating without issuing erroneous data at its output, but not without corrupting its internal state.

B. Circuit's Perspective

- *Inhibited transition:* In the most beneficial case the fault ties a signal to its current logic state and prohibits a further transition. Consequently, predecessor and/or successor stages endlessly wait for the inhibited transition, and no state corruption or incorrect output must be feared. While this can prevent the current handshake cycle from completing (immediate detection), it may as well be that a couple of computation cycles are (properly) finished without the need for the inhibited transition to take place, as in the case of a late detection.
- *Stimulated transition:* In the more adverse scenario the permanent fault causes an extra signal transition. This extra transition resembles the firing of a signal at an improper (too early) point in time, which is called *premature firing (PF)*. While DI logic is resistant against delays, i.e., late transitions, early transitions can do harm, by e.g., causing the capturing of invalid data. This is because they violate the causality chain of events.

Figure 1 summarizes and visualizes our analysis.

IV. Fault Injection Experiments Setup

A. Fault Model

The fault model that most accurately illustrates our targeted permanent faults is the *stuck-at fault (SAF)* model: a signal can be permanently stuck at a fixed logic value during system operation, either at logic high (stuck-at-1 or SA1) or at logic low (stuck-at-0 or SA0) [10]. We assume an input, or output, of a function block gets stuck rather than the whole net of a signal, which maps to the transistor getting defective and not the wires. Modeling the behavior of our system at a gate level, we further restrict ourselves to single faults.

B. Target Circuit

Our target pipeline is composed of 3 stages, with the one in the middle, S_2, representing our victim stage which we assume to be affected by permanent faults. S_1 represents the *upstream environment* of S_2 and is connected to an ideal data source, and S_3, the *downstream environment*, is connected to an ideal data sink. The circuit follows a 4-phase communication protocol with dual-rail one-hot data encoding (like NCL [11]) and completion detection to indicate the validity of data. Figure 2 shows the details of the circuit. It is based on the locally-opened normally-closed QDI return-to-zero latch from [12], where it is identified as robust against glitches.

For the logic function of our pipeline we chose a half adder, since adder structures have been considered representative in many other works as well [1], [6], [7], with a 2-bit datapath, to keep the simulation efforts in our validation manageable. It follows the delay-insensitive minterm synthesis (DIMS) implementation [13], [14]. This is a first attempt to represent *some* logic function, and it remains open for future work to analyze the influence of the logic function on the behavior.

Our register is formed by Muller C-elements (MCEs)[1], one for each signal rail: the control logic's output arms them for the expected next phase of the input, '1' for DATA and '0' for a NULL token.

Our completion detector (CD) is composed of an array of OR gates (two in our case) joining the two rails of each signal, and one MCE for joining the OR outputs.

As in [12], our latch consists of two completion detectors per stage, one left of the register (LCD) whose output switches

[1]The Muller C-element (MCE) is a fundamental gate in asynchronous circuits. It sets its output to high (low) if both its inputs are high (low). The output retains its value for non-matching inputs. It can be viewed as an AND gate with hysteresis.

from ‘0’ to ‘1’ to indicate valid NULL or DATA tokens, respectively, to be captured; and one right of the register (RCD) which generates an acknowledge signal to the upstream stage, to indicate whether the consumed token was NULL or DATA. The CDs feed the control logic unit – in our case another MCE. Its purpose is to trigger the register for capturing data when (1) the LCD indicates that the next DATA (NULL) token is available and (2) the ACK signal received from the downstream stage’s RCD indicates that the previous NULL (DATA) token has been processed and captured.

Finally, our source providing data to the input of S_1 is considered ideal, just characterized by the delay T_{src} between reception of ACK and generation of the next data token, whereas our ideal sink at the output of S_3 specifies the delay between reception of a data token and activation of ACK with T_{snk}.

C. Measures

Our experiments involve injecting faults into a VHDL model of the system shown in figure 2, that is executed in Mentor Graphics’ ModelSim HDL simulator. More specifically we used a behavioral simulation, however with a timing derived by the logical effort method [15]. This allows us to remain (largely) technology-independent. We injected faults into our victim stage S_2, systematically and exhaustively covering the whole space spanned by:

- Fault Polarity: We inject both, SA0 and SA1.
- Fault Location: We cover all inputs and outputs of the gates of the target stage, namely 22 locations in the half-adder, and 33 in the latch
- Speed of Source: To cover both a fast and slow source, we vary T_{src}.
- Speed of Sink: We apply the same variation to T_{snk}.
- Time of Injection: To make sure we hit all possible time intervals where a fault might have an effect, the injection time is incremented between experiments with a step smaller than the smallest gate delay used in our circuit. In this fashion, we have the same regular time grid for all experiments[2].

We only begin injecting a fault after the first token has entered the pipeline, and we always stop before the last one was sent in, to allow for the fault to manifest in the circuit.

For categorization of the fault effect we recorded the circuit’s observed reactions along the propagation path, using the following observations:

- Correctness of State: After a fault is injected, we let the system stabilize, then remove the fault and let the system run to completion. We use a second, identical but fault-free, pipeline as a golden reference for the correct behavior.
- Output Traces: We collect a trace of all signal transitions produced by the system at the sink and compare it with the trace of the golden reference.

[2]This means that for different settings of T_{src} and T_{snk} we get differing numbers of faults injected. This, however, corresponds well to reality, where longer phases are more prone to be affected by a fault.

- Pipeline Activity: We monitor whether the injection of the fault (a) creates an extra (non-causal) transition, (b) prevents the circuit from reaching the next phase, or (c) allows a move into the next phase.

For each injected fault we record event-traces of all these observations, along with the fault parameters, to a database and investigate the results by means of appropriate queries, to get detailed information about the internal dynamic behavior of the system.

We consider our experiments to have 2 different (essentially independent) dimensions: (1) fault effect (pipeline activity), (2) state corruption before deadlocking (correctness of state).

V. Results & Analysis

A. Possible Effects of a SAF in QDI Logic

For our analysis we identify the following possible effects: **immediate freeze (IF)** conveying when the pipeline freezes without moving to a subsequent phase, be it DATA after NULL or NULL after DATA. This still means that the circuit can be polluted in its respective current phase. **Late detection (LD)** signifies that some tokens keep moving forward in the pipeline before the circuit comes to a halt. **Premature firing (PF)** means the fault causes a transition to occur before it is scheduled and that it might have a malicious effect. In all cases, the circuit deadlocks and does not continue operation.

Figure 3 shows the rate of occurrence of each of the aforementioned effects. LD is the dominant effect ranging from 64.8% to 92.7%. IF and PF follow ranging from 7.3% to 25.1% and from less than 1% up to 10.1%, respectively. This constitutes the whole space: the pipeline always deadlocked and there were no other effects than anticipated.

Furthermore, figure 3 illustrates the effect of varying the delays for sink and source. As the source becomes faster than the sink ($T_{snk} > T_{src}$) (bubble-limited), the probability of the LD effect occurring gets higher. On the other hand, when the sink becomes faster ($T_{src} > T_{snk}$) (token-limited), IF and PF’s rates of occurrence increase.

B. State Pollution before Deadlock

Our next experiment attempted to quantify how many cases, with respect to the different effects, resulted in polluting the state of the circuit. As already mentioned above, we inspected the correctness of the system outputs when compared to their counterparts in the golden run. We evaluated this correctness by unfreezing the pipeline after it was given sufficient time for the fault to have manifested, by removing it and letting it run to completion. Table I shows the statistics of the incorrect cases that were observed during simulations, along with the distribution of the different effects. The former percentages, shown in the bottom row, ranged from 13.2%, to 29.8% (with 16.4% being the most occupied value) with respect to the total number of injections. All other effects are calculated on the basis of their contribution to this incorrect behavior.

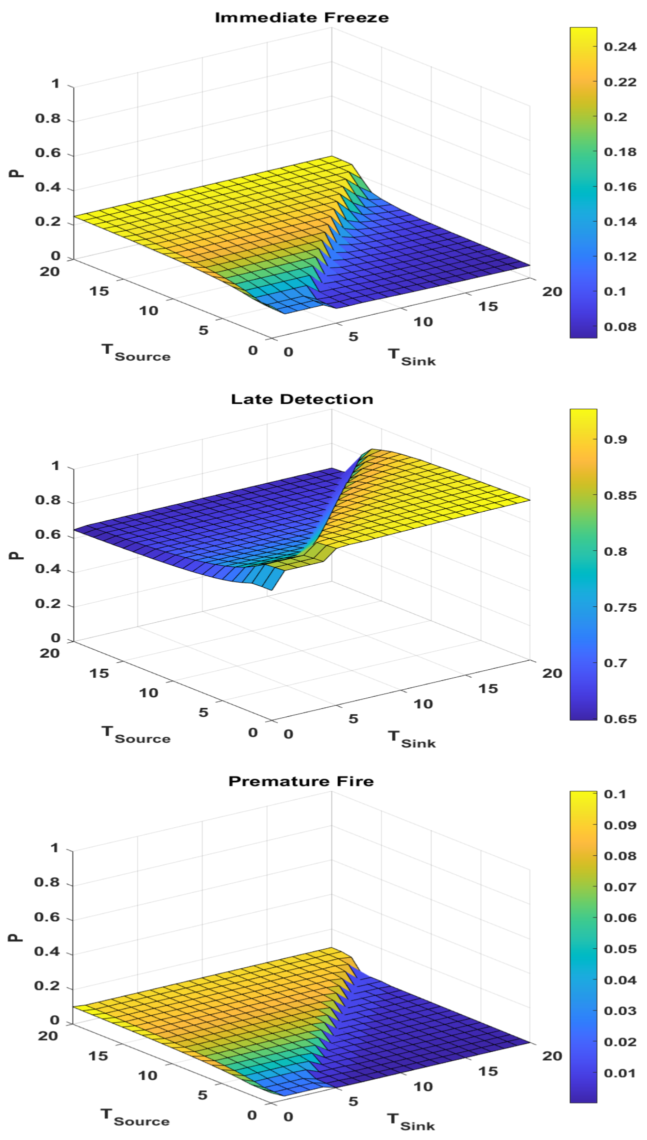

Fig. 3. Rate of Occurrence of Possible Effects

TABLE I
INCORRECT OPERATION AFTER UNFREEZE

Effect	**Incorrect Operation Count**		
	Minimum	***Average***	***Maximum***
Immediate Freeze	21.9%	38.5%	82.9%
Late Detection	10.2%	12.5%	29.5%
Premature Firing	1.4%	43.1%	97.0%
Incorrect State	13.2%	16.4%	29.8%

VI. Conclusion & Future Work

We have investigated the ability of QDI circuits to stop operation without state corruption in case of a permanent defect. This is an important feature for self-repair, as it saves the need for recovery and allows correctly continuing operation after removal of the defect. Our analysis has been based on injection of stuck-at faults in a simulation model of a 3-stage pipeline, with systematic coverage of the relevant parameter space. Not surprisingly, we could confirm that a QDI circuit always deadlocks if affected by a permanent fault. In addition to an immediate freezing of the operation, we have observed the effects of late detection, and premature firing known from the literature, and we have quantified their relative occurrence. It turns out that late detection is the dominant effect, which has a higher probability of occurring when the pipeline is bubble-limited; while immediate freezing is significantly less frequent, and the most undesired effect of premature firing is the most rare one, both having higher rates of occurrence when the pipeline is token-limited. Overall, we observed a proportion of 13% to 30% of faults (depending on the relative speed of source and sink) to corrupt the pipeline state in such a way that continuing operation after fault removal will lead to incorrect results. This is definitely a non-negligible share, and so additional provisions (or explicit recovery) will be necessary.

Our future work will extend the analysis to more complex target circuits, as well as further pipeline styles and protocols, while devising provisions to prevent erroneous transitions from propagating.

References

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

[2] W. Song, G. Zhang, and J. Garside, "On-line detection of the deadlocks caused by permanently faulty links in quasi-delay insensitive networks on chip," in *Proceedings of the 24th edition of the great lakes symposium on VLSI*, 2014, pp. 211–216.

[3] A. J. Martin and P. J. Hazewindus, "Testing delay-insensitive circuits," California Institute of Technology Pasadena Department of Computer Science, Tech. Rep., 1990.

[4] H. Hulgaard, S. M. Burns, and G. Borriello, "Testing asynchronous circuits: A survey," *Integration, the VLSI journal*, vol. 19, no. 3, pp. 111–131, 1995.

[5] S. J. Piestrak and T. Nanya, "Towards totally self-checking delay-insensitive systems," in *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*. IEEE, 1995, pp. 228–237.

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

[7] 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)*. IEEE, 2005, pp. 484–493.

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

[9] I. David, R. Ginosar, and M. Yoeli, "Self-timed is self-checking," *Journal of Electronic Testing*, vol. 6, no. 2, pp. 219–228, 1995.

[10] M. Yang, G. Hua, Y. Feng, and J. Gong, *Fault-tolerance Techniques for Spacecraft Control Computer*. Wiley Online Library, 2017.

[11] K. M. Fant and S. A. Brandt, "Null convention logic/sup tm: A complete and consistent logic for asynchronous digital circuit synthesis," in *Proceedings of International Conference on Application Specific Systems, Architectures and Processors: ASAP'96*. IEEE, 1996, pp. 261–273.

[12] W. J. Bainbridge and S. J. Salisbury, "Glitch sensitivity and defense of quasi delay-insensitive network-on-chip links," in *2009 15th IEEE Symposium on Asynchronous Circuits and Systems*. IEEE, 2009, pp. 35–44.

[13] D. E. Muller, "Asynchronous logics and application to information processing," *Switching Theory in Space Technology*, vol. 4, 1963.

[14] J. Spars and S. Furber, *Principles asynchronous circuit design*. Springer, 2002.

[15] I. Sutherland, R. F. Sproull, B. Sproull, and D. Harris, *Logical effort: designing fast CMOS circuits*. Morgan Kaufmann, 1999.