On the Reuse of Timing Resilient Architecture for Testing Path Delay Faults in Critical Paths

Felipe A. Kuentzer, Leonardo R. Juracy and Alexandre M. Amory
Faculty of Informatics, PUCRS University
Av. Ipiranga, 6681, Porto Alegre, Brazil
felipe.kuentzer@acad.pucrs.br, leonardo.juracy@acad.pucrs.br, alexandre.amory@pucrs.br

Abstract—**Energy efficiency has become one of the most common and important demands for contemporary applications, increasing the desire for chips that operate near the threshold voltage levels, which unfortunately worsens the effects of process, voltage, and temperature (PVT) variability. An alternative solution to cope with PVT variations are the timing resilient architectures, such as the synchronous Razor family and the asynchronous Blade template, that rely on error detection logic (EDL) to detect and recover from timing violations. On one hand, the use of timing resilient architectures makes the path delay testing more challenging because it is not a matter of simple pass or fails the test. On the other hand, we show that timing resilient architectures, such as Blade, present opportunities to design low-cost online delay testing of the critical paths. Results show the area overhead and fault coverage using functional testing on a 32-bit MIPS CPU and a crypto core.**

I. Introduction

As the VLSI design technologies approach more and more in the domain of low power, timing margins become a major factor when designing traditional synchronous circuits with optimal clock period considering variability in the manufacturing process, operation temperature, and supply voltage. Timing resilient architectures emerged as a promising solution to alleviate these worst-case timing margins by allowing timing errors, thus improving system performance by reducing frequency margins or energy consumption due to reduced voltage margins. On the other hand, these architectures need additional circuitry to detect and recover from these timing errors online.

Among timing resilient template architectures, Razor [1] detects errors through sequential circuits and restore valid data with one cycle penalty, while Razor II [2] is a simplification of Razor that uses architectural replay, requiring a complete pipeline flush, instead of using a specific recovery circuit. TIMBER [3] and [4] avoid architectural replay by borrowing time from neighbor pipeline stages. Bubble Razor [5] recovers from errors by stalling the pipeline. Timing resiliency was also applied to an asynchronous template. The asynchronous Blade [6] template recovers from timing errors by adding extra delay to the handshake communication with the following pipeline stages.

Despite the increasing evolution of resilient architectures, few papers address the testability of these templates. Testing timing resilient circuits are fundamentally different from testing conventional circuits because the former tolerates some timing violations, and not all timing violations become a fault. In addition, timing resilient circuits already add area overhead to detect violations, thus, the area for test circuitry is constrained. For example, when comparing Blade [6] and Bubble Razor [5] to a classical synchronous design, the overall area overhead for the Blade implementation is 8.4%, and 21% increase in combinational logic and 280% in the sequential area for the Bubble Razor implementation. To the second one, the area overhead of a design for testability can be prohibitive.

Testing of timing resilient circuits is essentially different from classic designs that target zero timing errors. On the other hand, the constant timing monitoring of resilient circuits provides feedback about the current state of the circuit during normal operation. Therefore, the resilient circuitry can be reused for testing. In this paper, we present an online delay test method for the Blade architecture. When a certain pipeline stage is in test mode, its critical paths are concurrently being tested for delay faults while the stage is still able to execute its normal operation. The only difference is that this stage is slower while it is in test mode. We demonstrate that the critical path selection is intrinsic to Blade's flow, and the number of paths covered by the test method is a trade-off with the area overhead. Besides, the fault detection mechanism relies on functional testing to exercise the critical paths during the test. The proposed modifications for online testing do not affect Blade's performance in normal mode and the silicon area overhead is below 1%. Fault coverage is presented for two case studies, a 3-stage MIPS CPU called Plasma [7] and an XTEA crypto core based on the Speed XTEA [8], both targeting an FDSOI 28nm technology. Lastly, we discuss how this proposal could also be applied to increase circuit lifetime and yield.

The remainder of this paper is organized as follows. Section II presents a background on testing timing resilient circuits and motivates this research. Section III reviews the Blade template and its most relevant aspects associated with this paper. Section IV presents the proposed test method and the necessary modifications to the Blade architecture, an analysis of performance and area impacts, and a discussion on possible uses of the proposed method for yield and lifetime improvements. Section V shows the results of silicon area and fault coverage for the two case studies. Finally, Section VI provides some conclusions.

II. Background and Related Works

From the technology perspective, there is no distinction between manufacturing asynchronous or synchronous CMOS circuits, which means that defects are similar in both designs [9], the same can be extended to synchronous or asynchronous timing resilient circuits. However, from the test perspective, timing resilient circuits are fundamentally different when compared to conventional circuits, since the former tolerates some timing violations, and not all timing violations become a fault. In this case, the test method must be capable of distinguishing a recoverable timing violation from an actual delay fault.

Despite this difference, a natural path when considering the testability of timing resilient circuits is to look at classical approaches used in the industry, such as the scan technique, which can be applied to various types of structural faults including stuck-at and delay. In [10], the authors propose the reuse of existing error tolerant circuitry of the Razor architecture to propose a new scan cell, called scan Razor flip-flop (SR-FF). With this new scan cell architecture the shift phase does not affect the inputs of the combinational logic, thus power dissipation during testing is reduced.

The work in [11] starts with a classical mux-D scan cell, that is further modified to act as a timing violation detector and recovery mechanism. The Time Dilation scan architecture is suitable for online (concurrent) timing error detection and recovery and supports classical off-line scan testing. The idea of reusing error detection circuitry for online test purpose is developed in [12], where a scan-based aging monitoring scheme is proposed. The circuit is monitored during normal operation, so no faster-than speed testing is applied, and it gives an alarm if aging is detected so that actions can be taken before a major failure. The experimental results show that the proposed solution consumes less power and presents less overhead in large designs when compared to previous methods. Unfortunately, the authors do not provide additional information about these results.

The test methodology presented in [13] [14] is, as far as we know, the first method to address the main differences between testing conventional synchronous circuits and testing timing resilient architectures. The authors proposed a scan architecture based on the Double Sampling With Time Borrowing (DSTB) design presented in [15]. They included a clock divider and a duty-cycle control circuit to select the test conditions used to differentiate timing violations from delay faults in the combinational logic. The error detection circuitry from the resilient architecture is reused to detect manufacturing faults. The classic two-vector test stimuli used in delay fault testing is applied using the proposed scan DSTB cell. As presented in this current paper, we believe that an asynchronous timing resilient approach can present a better solution to cope with PVT and reduced power consumption [6], and the approach proposed in [13] may not be entirely applicable for testing Blade, as it is further described in Section IV to introduce our proposal.

Finally, asynchronous circuits have some advantages over synchronous regarding its testability. For instance, a stuck-at fault in the asynchronous controller can halt the entire circuit, making it easy to detect such fault. In [16], a fault analysis of the Blade's EDL [6] is presented. It is discussed how the resilient circuitry can be reused for detecting stuck-at and delay faults inside the EDL. It also shows a fault classification method based on the behaviors observed in the faulty EDLs, such as the circuit halt.

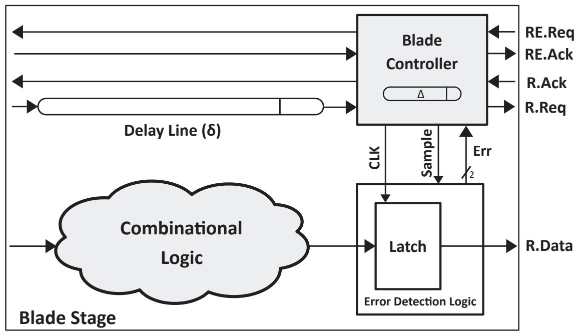


Fig. 1: Blade template [6].

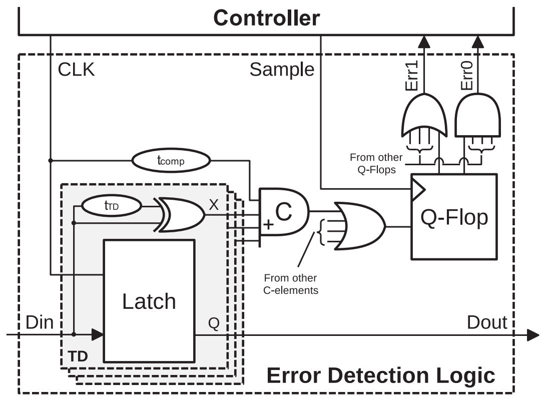
III. Blade Template Overview and Timing Analisys

Blade [6] is an asynchronous design style that was proposed to overcome metastability issues in many of the proposed timing resilient architectures, reported in [17], and also to reduce the high timing error penalties originated by the recovering mechanism of resilient architectures. Fig. 1 illustrates the basic Blade architecture. It consists of an asynchronous Blade controller, two delay lines, and an EDL. The Blade controller communicates with other stages using a typical bundle-data channel L/R. The δ delay controls the moment that data at the output of the combinational logic can be sampled and propagated through the latch. The Δ delay defines the amount of time that the latch is transparent, and is defined as the timing resiliency window (TRW). A timing violation is flagged if data changes during the TRW. The delay values of δ and Δ must be designed to be sufficiently large to cover the longest critical path in its corresponding pipeline stage.

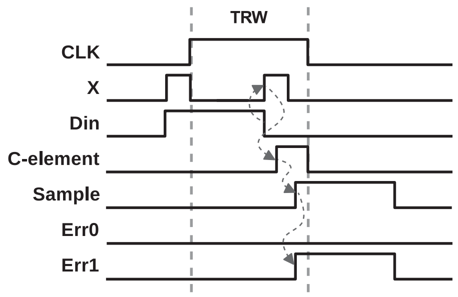
The error detection logic flags a timing violation by asserting its *Err* signal. The Blade controller then communicates with its right neighbor using an additional error channel RE/LE. To recover from the timing violation the next stage delays by Δ the latch opening, until the correct data is propagated through the combinational logic.

Fig. 2(a) shows the EDL used in Blade. As detailed in [6], the design consists of error detecting latches, Q-Flops [18] and an asymmetric C-elements. The C-element acts as a memory cell that stores any violation detected during the high phase of the *CLK*. The C-element switches to 0 if *CLK* is at 0 and to 1 only if both *CLK* and the *X* output of the transition detector (TD) is at 1.

The output of the C-element is sampled at the end of the TRW by the Q-Flop. The Q-Flop ensures safe operation against metastability by an internal filter, and its outputs only change if data is not metastable. The dual-rail signal *Err*,



(a)



(b)

Fig. 2: (a) EDL architecture [6]. (b) Timing diagram of the EDL error detection.

composed by wires *Err0* and *Err1*, stalls the controller until the outputs are stable and it can safely evaluate if an error occurred. The delay element t_{TD} defines the transition detector pulse width, while t_{comp} is the compensation delay added to ensure that a transition before the rising edge of *CLK* is not flagged as a violation.

Fig. 2(b) shows the timing diagram of an error being detected. The propagation delay of some internal cells are omitted, and to simplify further analysis the TRW is equal to Δ delay. During the TRW the *Din* signal falls. The TD then flags the transition with *X*, which is stored by the C-element until the end of the TRW. Right before the end of the TRW, *Sample* activates the Q-Flop that informs to the controller through *Err1* that a timing violation was detected.

The Blade controller implements a new form of asynchronous handshaking protocol, called *speculative handshaking* [6]. The implementation is divided into three interacting Burst-Mode state machines [19], and the complete description is available in the original paper. In this paper, we focus only on the description of the *CLK* output circuit of Fig. 3(a), that is modified for the test purpose. In the original implementation, the controller generates an internal clock signal (*int_clk*) that activates the *CLK* output and an internal delay line. After the signal is propagated through the delay line the *delay* signal indicates to the controller that the *CLK* signal must be deactivated. This behavior can be seen in Fig. 3(b).

Blade's automated flow uses industry standard tools, including DesignCompiler and PrimeTime from Synopsys and NC-Sim from Cadence. The flow converts a single clocked synchronous RTL design into an asynchronous Blade design. There are 5 main steps for the circuit conversion: *Synchronous Synthesis*, *Flip-flop to Latch*, *Retiming*, *Resynthesis* and *Blade Conversion*. The synchronous flop-based design is converted to a latch based design. During the retiming and resynthesis phase, the circuit is optimized so that pipeline stages are balanced, and then a list of critical paths is created. With this, only latches in critical paths are replaced by EDLs, thus reducing the area overheads originated by the Blade conversion. The number of critical paths is determined by the amount of timing margin removed. As more timing margin is removed, more critical paths are replaced by Blade's EDL. At the end, a co-simulation environment is used to functionally validate the implementation. The co-simulation environment consists of a testbench where a stream of inputs is forked to both the synchronous and Blade netlists, and their stream of outputs are compared.

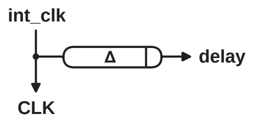
IV. Proposed Test Method

Timing resilient circuits suffer from area overheads that come from the error detection and recovery mechanism, and this overhead can limit the extra silicon area for the testing circuitry. The reuse of error detection circuit as a fault detection mechanism is a promising approach to reduce the DfT area overhead. In [13], the resiliency circuitry is modified to create a scan chain for testing the datapath. Specifically for path delay testing, the error detection circuit is responsible for detecting a delay fault in the critical paths. Instead of scanning out patterns after each test through the datapath scan chain, an additional scan chain is used for capturing only the error signals generated by the error detection circuits.

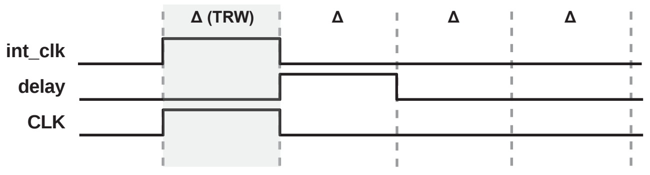
The different test operations are created by changing the clock duty-cycle, making the circuit to operate slower or faster than its normal operation. It would be ideal to apply this technique for testing Blade, but as an asynchronous architecture, the absence of a global clock would require additional circuitry in order to create a synchronous mode of operation during the test mode. Additional clock control circuitry and scan chain, used only for testing, is a cost that we want to avoid.

Therefore, the proposed test method explores the concurrent test capabilities of circuits implemented with the asynchronous Blade by the reuse of the EDL as a fault detection mechanism. It also relies on functional testing to produce test patterns for the critical paths, reducing the area overhead required by a dedicated scan chain.

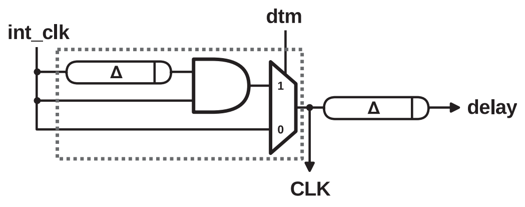
As presented in [20], functional testing may translate into delay fault under-testing, where non-functional delay paths are not covered. On the other hand, structural testing can overtest paths that are not functionally activated, resulting in yield loss. We argue that if the defect does not affect performance in the normal functional mode of operation, a missed defect should not cause a system failure. In this case, the chip should not be rejected based on a failure in a non-functional path. Specifically for testing delay faults in critical paths in Blade,



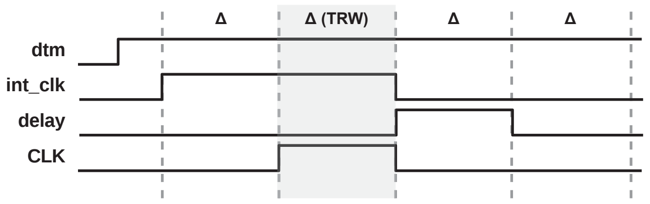
(a)



(b)



(c)



(d)

Fig. 3: (a) Original CLK output circuit. (b) Timing diagram of original circuit. (c) Modified CLK output circuit. (d) Timing diagram of modified circuit.

we believe that functional testing can achieve a satisfactory fault coverage with a low test circuitry overhead. This is explored in the proposed test method, including its use for yield increase and aging monitoring.

The proposed test approach focus on online testing of path delay faults of critical paths as a natural consequence of Blade's implementation, where critical paths have an EDL constantly monitoring possible timing violations. In order to detect faults through the EDL, the Blade controller is modified to shift the TRW, which essentially means, to make the affected pipeline stages to operate in a slower mode. Note that this slower mode is only applied for testing, and the modifications do not affect the normal mode of operation. Now the EDL will detect timing violations later than the normal operation. These later timing violations are in fact the delay faults being detected, that otherwise would not be captured.

Fig. 3(c) shows the modified circuit that creates the shifted TRW. The AND gate before the multiplexer ensures the *CLK* deactivation immediately after the controller deactivates the *int_clk*, otherwise, the TRW would also be extended by the additional Δ delay. The *dtm* (delay test mode) signal selects whether the TRW must be shifted or not. The timing diagram of Fig. 3(d) illustrates this behavior. The rest of the controller circuit remains unchanged, and the modifications are transparent to the rest of the circuit since the controller still waits for the *delay* signal transition to continue with the handshake protocol. Timing constraints and the performance in normal mode are not affected either, and the delays of the additional gates can be compensated in the delay line. Note that, the method is limited to detect faults that do not exceed the additional Δ time, and faults that trespass the shifted TRW are not captured. Another thing to mention is that the proposed test method does not consider faults produced by small delay defects, only large delays. This will be addressed in future works.

An *error_o* pin is added to observe externally the faults detected by an EDL. As shown in Fig. 4, the output of an OR gate is connected to this new pin. The OR gate groups the *Err1* signals from all the EDLs on the circuit. Once the *dtm* is enabled, the *error_o* is constantly monitored while the circuit executes a functional test. If at any moment a transition at *error_o* occurs, a delay fault is considered detected.

A. Discussion on the test method enhancement

The proposal evaluated in this paper assumes that the TRW shift is implemented with an additional delay line. This delay line is the one inside the dashed region of Fig. 3(c). However, it is also possible to have an alternative design where, for instance, there are multiples activations of a single delay line. Another assumption is that, a single primary input is connected to all internal *dtm* signals, and it affects all controllers. In this case, all controllers will shift the TRW at the same time. Alternatively, the *dtm* signal of each controller can be controlled individually by using a scan chain as presented in Fig. 4. The dashed *global_dtm_i* is replaced by an auxiliary scan chain connected to *dtm_i*.

This alternative design opens some new possibilities in terms of fault diagnose, yield improvement and aging monitoring. For instance, assume that there is a delay fault in a critical path between controllers C1 and C2. If the *global_dtm_i* approach is applied, there is no way to determine, just by observing *error_o*, which stage contains the fault. If the *dtm* of each controller can be set individually, a diagnosis procedure can be executed to discover which stage captured the fault. For this example, only the DTM2 scan register will hold an enable for the C2 controller to shift its TRW.

This implementation allows an increase in yield if the fault is detected during manufacturing test and the shifted TRW is able to identify the delay fault as a timing violation. Thus, by individually controlling each controller will shift the TRW, it is possible to enable permanently the *dtm* of stages where faults were detected, even during normal operation. The consequence is a stage that is Δ time slower than its original design. Moreover, with the advantage of being an asynchronous design, only the faulty stages can be slowed down, and the overall performance impact will not be like in a synchronous design, where all stages would be affected by a clock frequency reduction that accounts only for the faulty stages. In this case, a circuit that would be discarded can still be commercialized as a lower performance version.

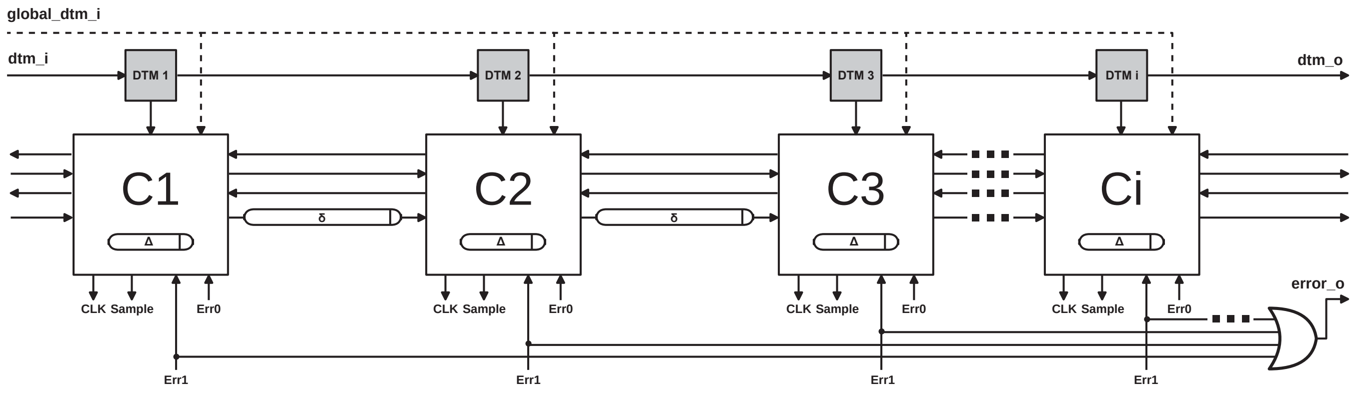


Fig. 4: Alternative design with individual control over the dtm signal.

Still, on the new possibilities of this implementation, an aging monitoring mechanism can be applied to detect a performance degradation during the circuit lifetime by monitoring the *error_o* signal. The constant monitoring of this signal can be translated into a performance metric, the error-rate. An increase in the error-rate can be related to circuit aging, and actions can be taken to avoid the circuit failure, increasing the circuit lifetime.

Once the aging is detected, a diagnose test, similar to the one described earlier, can be performed in order to detect which stages are being affected. Also, like the solution to increase yield, the affected stages could be purposely set to operate in a slow mode by enabling their delay test mode during normal operation. Improvement in yield and aging monitoring will be further explored in future papers.

V. Experiments and Results

The proposed test method was included into Blade's automated flow, automatically adding *dtm* port, instantiating the OR gate to group the *Err1* signals and making their proper interconnections. The flow targets a 28nm FDSOI technology.

The proposed test method is evaluated with a 32-bit XTEA crypto core, based on the Speed XTEA described in [8], and to compare area results with the original Blade proposal, the same case study presented in [6] was implemented, a 3-stage 32-bit MIPS OpenCore CPU called Plasma [7].

The fault coverage results are extracted by a delay fault simulation incorporated into Blade's co-simulation environment. The fault simulation environment has the following input parameters: a list of critical paths extracted from the Blade flow, where the EDL is inserted and the delay fault is injected; the Δ delay; and the original SDF (Standard Delay Format) file. For each critical path, a mutant SDF is generated with a timing violation for that specific path. The additional time is equivalent to the Δ delay plus the slack of the path. This adds a propagation delay that shifts the path transitions to the shifted TRW, thus if any transition occurs in this path, the transition will be flagged by the EDL as delay fault. After all critical paths are simulated, the fault simulation environment presents a report with the total simulated paths and the fault coverage.

TABLE I: Fault coverage for critical paths of Bladed Plasma and XTEA core.

	Plasma	**XTEA**
Total Paths	625	6271
Critical Paths	238	937
Detected Faults	172	937
Fault Coverage	72.27%	100%

Table I presents the fault coverage for both case studies. For the XTEA fault simulation, a testbench randomly generates data to the netlist inputs. The simulation environment shows that 100% of the delay faults in the critical paths were detected.

The experiments with Plasma used the testbench provided with the OpenCore package to produce the input data. Reports for Plasma fault simulation show a 72.27% fault coverage. This lower fault coverage observed with Plasma is explained by the use of functional code that was not developed aiming functional testing. Most of the uncovered paths are related to high-order bits from registers that had no transition during the simulation, such as the *program counter* and *memory address* registers. As already pointed, a transition must occur in the path so the EDL captures the delay fault. Although this is out of the scope of this paper, software-based testing can be used in order to improve fault coverage of processors [21] [22]. For example, the work in [23] used Plasma as a case study and reported 95% of fault coverage.

Table II shows that for both case studies have area overhead lower than 1%. The additional area comes from the increment in the Combinational logic and the Buf/Inv, which is the additional delay line. This can be further reduced by reusing the existing delay line instead of creating a new one. Moreover, the area overhead of the proposed test circuitry does not scales up with the number of critical paths covered. If more paths are selected in the Blade flow to receive an EDL, the area overhead comes only from the additional EDL circuitry. This way, the designer can tradeoff between additional EDL area

TABLE II: Comparison of Bladed Plasma vs Bladed Plasma-DTM (PDTM) and Bladed XTEA vs Bladed XTEA-DTM (XDTM) in terms of area (μm^2).

	Plasma	**PDTM**	**XTEA**	**XDTM**
Combinational	7095.28	7124.33	35561.44	35599.80
Buf/Inv	608.90	630.93	2959.80	2991.13
Noncombinational	7860.69	7860.69	21828.00	21828.00
Macro/Black Box	228.58	228.58	1280.01	1280.01
Net Interconnect	undefined	undefined	undefined	undefined
Total Area	15793.45	15844.53	61629.25	61698.94
Area Overhead	-	**0.32%**	-	**0.11%**

and the number of paths covered by timing resilience and the proposed test method.

VI. Conclusion

This paper presents a concurrent test approach for detecting delay faults in critical paths with the Blade timing resilient architecture. This test method takes advantage of the Blade's error detection circuit and, by making a small modification into the Blade controller, it becomes an online delay fault detection mechanism. The additional test circuitry does not affect the original circuit performance and functionality under normal operation and has an area overhead lower than 1%. Under the delay test mode, the circuit flags timing violations that exceed the timing resiliency, and would not be captured under normal operation. Results for fault coverage show that the functional stimuli executed during test plays an important role in the method. For future works, the test method analysis can be extended to small delay defects and also the test circuitry can be integrated into a yield improvement technique and an aging monitoring approach.

Acknowledgment

The authors would like to thank Peter A. Beerel and his research group for the help with the Blade's flow and discussions about testability issues. Alexandre would like to thank CNPq for the financial support (process 460205/2014-5).

References

[1] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a low-power pipeline based on circuit-level timing speculation," in *36th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2003, pp. 7–18.

[2] D. Blaauw, S. Kalaiselvan, K. Lai, W. H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor II: In situ error detection and correction for PVT and SER tolerance," in *IEEE International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC)*, Feb 2008, pp. 400–622.

[3] M. R. Choudhury, V. Chandra, R. C. Aitken, and K. Mohanram, "Time-borrowing circuit designs and hardware prototyping for timing error resilience," *IEEE Transactions on Computers (TC)*, vol. 63, no. 2, pp. 497–509, Feb 2014.

[4] S. Kim and M. Seok, "Variation-tolerant, ultra-low-voltage microprocessor with a low-overhead, within-a-cycle in-situ timing-error detection and correction technique," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 50, no. 6, pp. 1478–1490, Jun 2015.

[5] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. M. Harris, D. Blaauw, and D. Sylvester, "Bubble Razor: Eliminating timing margins in an ARM Cortex-M3 processor in 45 nm CMOS using architecturally independent error detection and correction," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 48, no. 1, pp. 66–81, Jan 2013.

[6] D. Hand, M. T. Moreira, H. H. Huang, D. Chen, F. Butzke, Z. Li, M. Gibiluka, M. Breuer, N. L. V. Calazans, and P. A. Beerel, "Blade – A timing violation resilient asynchronous template," in *21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2015, pp. 21–28.

[7] "Plasma cpu," http://opencores.org/project,plasma, 2014.

[8] J.-P. Kaps, *Chai-Tea, Cryptographic Hardware Implementations of xTEA*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 363–375.

[9] M. Roncken, "Defect-oriented testability for asynchronous ICs," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 363–375, Feb 1999.

[10] A. Anastasiou, Y. Tsiatouhas, and A. Arapoyanni, "On the reuse of existing error tolerance circuitry for low power scan testing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 1578–1581.

[11] S. Valadimas, A. Floros, Y. Tsiatouhas, A. Arapoyanni, and X. Kavousianos, "The time dilation technique for timing error tolerance," *IEEE Transactions on Computers (TC)*, vol. 63, no. 5, pp. 1277–1286, May 2014.

[12] H. Yi, T. Yoneda, and M. Inoue, "A scan-based on-line aging monitoring scheme," *Journal of Semiconductor Technology and Science (JSTS)*, vol. 14, no. 1, p. 124–130, Feb 2014.

[13] F. Yuan, Y. Liu, W.-B. Jone, and Q. Xu, "On testing timing-speculative circuits," in *50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–6.

[14] Q. Han, J. Guo, W. B. Jone, and Q. Xu, "Path delay testing in resilient system," in *56th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug 2013, pp. 645–648.

[15] K. A. Bowman, J. W. Tschanz, N. S. Kim, J. C. Lee, C. B. Wilkerson, S. L. L. Lu, T. Karnik, and V. K. De, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 44, no. 1, pp. 49–63, Jan 2009.

[16] F. A. Kuentzer and A. M. Amory, "Fault classification of the error detection logic in the blade resilient template," in *22nd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2016, pp. 37–42.

[17] S. Beer, M. Cannizzaro, J. Cortadella, R. Ginosar, and L. Lavagno, "Metastability in better-than-worst-case designs," in *20th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2014, pp. 101–102.

[18] F. U. Rosenberger, C. E. Molnar, T. J. Chaney, and T. P. Fang, "Q-modules: internally clocked delay-insensitive modules," *IEEE Transactions on Computers (TC)*, vol. 37, no. 9, pp. 1005–1018, Sep 1988.

[19] R. M. Fuhrer, B. Lin, and S. M. Nowick, "Symbolic hazard-free minimization and encoding of asynchronous finite state machines," in *IEEE International Conference on Computer Aided Design (ICCAD)*, Nov 1995, pp. 604–611.

[20] A. Krstic, J.-J. Liou, K.-T. Cheng, and L. C. Wang, "On structural vs. functional testing for delay faults," in *4th International Symposium on Quality Electronic Design (ISQED)*, Mar 2003, pp. 438–441.

[21] W.-C. Lai, A. Krstic, and K.-T. Cheng, "Test program synthesis for path delay faults in microprocessor cores," in *IEEE International Test Conference (ITC)*, 2000, pp. 1080–1089.

[22] V. Singh, M. Inoue, K. K. Saluja, and H. Fujiwara, "Instruction-based self-testing of delay faults in pipelined processors," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 14, no. 11, pp. 1203–1215, Nov 2006.

[23] N. Kranitis, A. Paschalis, D. Gizopoulos, and G. Xenoulis, "Software-based self-testing of embedded processors," *IEEE Transactions on Computers (TC)*, vol. 54, no. 4, pp. 461–475, Apr 2005.