Testable Error Detection Logic Design Applied to an Asynchronous Timing Resilient Template

Felipe A. Kuentzer, Leonardo R. Juracy, Matheus T. Moreira* and Alexandre M. Amory
Faculty of Informatics, PUCRS University
*Chronos Tech, Research and Development
felipe.kuentzer@acad.pucrs.br, leonardo.juracy@acad.pucrs.br, matheus@chronostech.com, alexandre.amory@pucrs.br

***Abstract*—Resilient circuits are becoming a popular alternative to cope with process, voltage, and temperature variability under ultra-deep-submicron technology. Timing resilient architectures rely on error detection logic (EDL) to detect and recover from timing violations. Different EDLs have been proposed to either reduce the area overheads associated with the additional circuitry or to reduce recovery time, but most of them do not account for testability. This paper proposes a testable EDL (TEDL) architecture for manufacturing and field testing. Fault coverage and area overhead are illustrated on a resilient implementation of Plasma, a 3-stage OpenCore MIPS CPU, which contains the proposed testable EDL circuitry. The results show that 100% of the stuck-at faults of the TEDL are detectable with 4.61% area overhead when compared to the Plasma with the original EDL design.**

***Index Terms*—timing resilient, error detection logic (EDL), asynchronous design, design for testability (DfT), stuck-at fault.**

I. Introduction

Energy efficiency has become one of the most common and important demands for contemporary applications, increasing the demand for integrated circuits that operate near the threshold voltage levels, which unfortunately aggravates the effects process, voltage, and temperature (PVT) variability. Traditionally, timing margins are incorporated to the clock period in order to compensate for the uncertainties of PVT variations, and translate to performance, power and area losses. Timing resilient architectures emerged as a promising solution to alleviate these margins, improving system performance while reducing energy consumption by allowing operation at reduced voltages. These architectures need additional circuitry to detect and recover from timing violations.

Timing resilient architectures include Razor [1], which detects timing violations using an EDL and corrects valid data through a dedicated recovery circuit with one clock cycle penalty. Razor II [2] is a simplification of Razor that uses architectural replay, requiring a complete pipeline flush, instead of using a specific recovery circuit. Bubble Razor [3] recovers from errors by stalling the pipeline in order to avoid the need of architectural replay, thus reducing the performance penalties associated with error recovery. TIMBER [4] and [5] avoid architectural replay by borrowing time from neighbor pipeline stages. Timing resiliency was also applied to asynchronous circuits in Blade [6] and SafeRazor [7]. Both proposals present solutions to metastability of previous designs [8]. In particular, Blade recovers from timing errors by adding extra delay to the handshake communication to the following pipeline stages.

Resilient architectures rely on error detecting sequential (EDS), to detect when a setup timing violation occurs. To convert a traditional design into a timing resilient one, the sequential elements are replaced by these EDSs, thus an increment in area and power is associated with the additional detection mechanism. For example, when comparing Blade [6] and Bubble Razor [3] to a traditional synchronous design, the overall area overhead for the Blade implementation is 8.4% and Bubble Razor presents 21% increase in combinational logic and 280% in the sequential area. To account for these overheads, new power and area efficient EDS elements where proposed in [9] and [10].

Despite the increasing evolution of resilient architectures, little has been done regarding the testability of these designs. A problem when considering the testability of timing resilient architectures is the area overhead of test circuitry since the resilient implementation already incurs in significant area overhead. Moreover, some EDS proposals use custom sequential cells that are not available in most technology libraries, thus it is not possible to take advantage of automated design for testability (DfT) flow with commercial Electronic Design Automation (EDA) and Automatic Test Pattern Generation (ATPG) tools. This paper demonstrates the challenges of using ATPG tools with the Blade resilient architecture and presents a testable EDL (TEDL) that is compatible with classic DfT techniques available in commercial EDA tools. The proposed architecture is compared to the original EDL presented in [6] using a 3-stage MIPS CPU called Plasma [11], targeting an FDSOI 28nm technology. Experimental results show that the proposed TEDL has 100% of single stuck-at fault coverage, with about 4.61% of area overhead compared to the Plasma with the original Blade's EDL.

The remainder of this paper is organized as follows. Section II presents related DfT designs for testing timing resilient circuits. Section III introduces Blade's original EDL. Section IV discusses a preliminary fault coverage analysis of Blade's EDL using classic ATPG approach and a behavioral approach. Section V describes the proposed testable EDL architecture along with the test methodology. Section VI shows the case study and the results for fault coverage and area overhead.

978-1-5386-7431-4/18/$31.00 ©2018 IEEE

Finally, Section VII provides some conclusions.

II. Related Works

When considering the testability of timing resilient circuits, previous works look at classic approaches used in industry, such as the scan technique, which can be applied to various types of structural faults including stuck-at and delay. Anastasiou et al. [12] proposes 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.

Another similar approach is the Time Dilation scan architecture [13], where a classic mux-D scan is modified to act as a timing violation detector and recovery mechanism. The Time Dilation scan architecture is suitable for online timing error detection and recovery and supports classical off-line scan testing. Results show a reduction in area and power consumption when compared to the Razor flip-flop (R-FF), but do not address the testability of the resilient circuit itself.

Yuan et al. [14] present a scan architecture based on the Double Sampling With Time Borrowing (DSTB) EDS, proposed by [15]. In this approach, the error detection circuitry from the resilient architecture is reused to detect manufacturing faults. A similar approach has been applied to Blade [16], where the error detection logic is applied to help to detect and diagnose its internal faults. Differently from the previous approaches, no scan technique is applied to the EDS. A fault classification method is also presented. However, instead of classic ATPG, the method relates the overall behavior of the circuit to an existing fault in the EDL, such as the circuit halt. Results show that Blade's original EDL has low stuck-at fault coverage (30%), and a single fault in the EDL can completely disable its ability to detect timing violations. More details will be presented in Section IV-B.

III. Blade's Error Detection Logic

The notion of resiliency has been proposed by different authors in the literature with different error detection circuits. They include a shadow latch, proposed in [3], that captures the data before the main latch to compare and generate an error signal and a side-channel error detection strategy [17] that takes advantage of an inherent redundancy in a standard flip-flop cell. This paper focuses on the Blade template [6] and its EDL, illustrated in Figure 1.

Blade's EDL [6] consists of error detecting latches with a transition detector, that is based on the Transition Detection With Time Borrowing (TDTB) EDS [15], asymmetric C-elements and Q-Flops [18]. The transition detector is based on an XOR function of the data input and a delayed version of this input, which produces a pulse at *X*, thus indicating a data transition. 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 a TD is at 1. The output of the

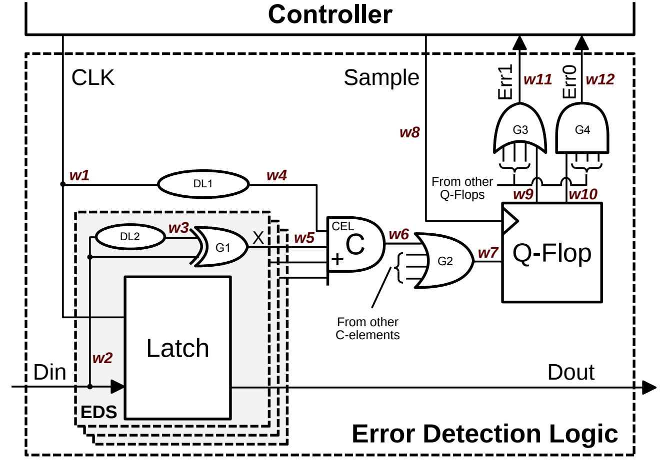

Fig. 1: Original EDL diagram [6], including nets named from *w1* to *w12*.

C-element is sampled by the Q-Flop at the end of the *CLK* high phase.

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*, 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 *DL2* defines the transition detector pulse width, while *DL1* is the compensation delay added to ensure that a transition before the rising edge of *CLK* is not flagged as a violation. The other logic elements, such as *G2*, are designed to amortize the area overhead of the C-elements and Q-Flops across multiple pipeline stages.

IV. Preliminary Testability Analysis of Blade's EDL

Before presenting the proposed method in the next section, this section discusses the testability issues of Blade's EDL using two different strategies to perform fault analysis.

A. ATPG Fault Coverage Analysis

In this Section, the stuck-at fault coverage for the original EDL is reported using Synopsys TetraMAX ATPG tool. For this particular analysis, a netlist targeting 28nm FDSOI technology is generated with DesignCompiler from Synopsys. The netlist consists of a two-stage Blade pipeline (Figure 2), where the EDSs of the first stage are directly connected to the ones of the second stage. This scenario simplifies the integration and analysis with Synopsys tools since it removes the particularities of an asynchronous resilient template. Instead of asynchronous controllers, non-overlaping clock signals are created, one for each stage, and the circuit becomes essentially a synchronous latch based design, and all primary inputs are controllable and all primary outputs are observable.

Blade has unconventional sequential cells such as the transition detector, the C-element and the Q-Flop which are not typically available in most standard-cell libraries. Moreover, their correspondent scan cells are either not available in most libraries or not supported by comercial DfT flows. Thus, it

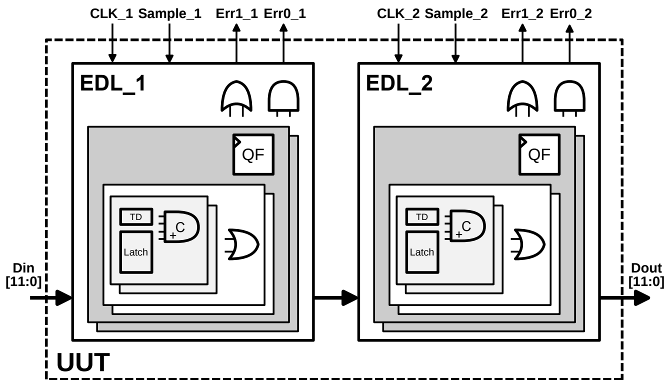

Fig. 2: ATPG test scenario. The transition detectors (TD) are connected to three-input C-elements (C) and the C-elemets to two-input OR gates. In total there are 6 TDs to each OR gate. The OR gate is then connected to a Q-Flop (QF). With 2 QF per step, each step is 12 bits wide.

would not be possible to use automated tools to obtain the fault coverage reports. In order to overcome this problem, without compromising the applicability of our solution, those unconventional sequential cells were modeled as macro blocks consisting of only cells already supported by the 28nm FD-SOI library, including latches and flip-flops, to implement the sequential behavior. An advantage of this approach is that Blade's EDL can be evaluated with or without a scan chain because the sequential cells (latches and flip-flops) are recognized by the DfT tools to perform automatic scan chain generation. The asymmetric C-element behavior is described in a similar way to [19]. The Q-Flop is described as a standard flop that is reset when its enable is low, which is, in terms of behavior, equivalent to the Q-Flop. Finally, the resulting netlist is functionally validated before starting the DfT flow.

The fault simulations performed by the ATPG tool considered only single stuck-at faults. The tool is configured to ignore all the first stage, redundant wires, and internal nodes from the macro cells, which gives a total of 12 fault points. These 12 points are the labeled wires of Fig. 1, named *w1* to *w12*. The fault coverage of the Blade‘s original EDL is evaluated considering two scenarios: using no scan structure and replacing all latches by the Level Sensitive Scan Cells [20].

Table I shows the summary reported by TetraMAX. The faults are classified into five classes and the fault coverage is calculated by the total number of faults divided by the number of Detected plus the Possibly detected faults. The low fault coverage is related to the lack of observability and controllability of internal nodes. Even adding a scan cell is not enough to improve the fault coverage. This is mainly caused by the C-element and the Q-Flop, that, due to their functional behavior, do not allow the tool to stimulate and observe faults in their path.

A possible solution is to make the C-element scannable. Scannable C-elements were proposed in the past, such as in [21] and [22]. These solutions are not considered in this paper because they present a higher impact in area and performance as it is further discussed later. In the case of the Q-Flop, the problem lies on guaranteeing that the metastability filter is not affected by the modifications in the cell design. The second issue is that commercial DfT tools only recognize conventional sequential cells, such as latches or flip-flops. Thus, the designer would not be able to use automated scan chain generation for the C-element and the Q-flop.

TABLE I: TetraMAX stuck-at fault summary report.

	no-Scan	Scan
Detected	6	7
Possibly detected	2	1
Undetectable	1	1
ATPG untestable	0	0
Not detected	15	15
not-controlled	(7)	(7)
not-observed	(8)	(8)
Total Faults	24	24
Fault Coverage	33.3%	33.3%

B. Behavioral Fault Coverage Analysis

A complete stuck-at fault analysis of Blade's original EDL is presented in [16]. This work follows a different approach from the one presented in Subsection IV-A. Instead of relying on classic ATPG approach, faults are detected through a fault classification method based on the functional behavior of the entire EDL when in the presence of an internal fault.

A similar concurrent test approach has been used for a network-on-chip [23] and a processor [24]. Chrysanthou et al. [23] use a group of lightweight micro-checker modules in a network-on-chip as concurrent hardware assertions, checking for illegal output patterns while the circuit is in normal mode. This approach provides full fault coverage and diagnostic capability. Wang and Patel [24] propose the use of a symptom-based error detection approach to detect atypical events that concurrently hint the occurrence of soft errors in a processor. These events trigger a rollback to a safe checkpoint, restoring the processor state.

It has been demonstrated in [16] that a single stuck-at fault might jeopardize the resilience of part of the circuit. The fault coverage analysis considered three test approaches. The first one relied only on functional testing of the circuit, where most of the faults were detected through a pipeline halt observation, accounting for 34% of fault coverage, which provides a similar fault coverage compared to the classic ATPG approach. The second approach presented in [16] considered the use of a scan chain to make the signals *Err0* and *Err1* externally observable, which increased the fault coverage to 66%. It has been observed that the undetected faults were activated only when a timing violation (TV) was detected. Thus, in order to achieve 100% of fault coverage, a successful test method must be able to inject a TV during the test. It was shown that it is required to inject TV in test mode, but it does not show how to do it. Also, it was observed that some faults were only detectable by observing the EDL's error signals of the next stage, and not the signals of the faulty EDL. We observed that when a fault prevents the EDL from detecting

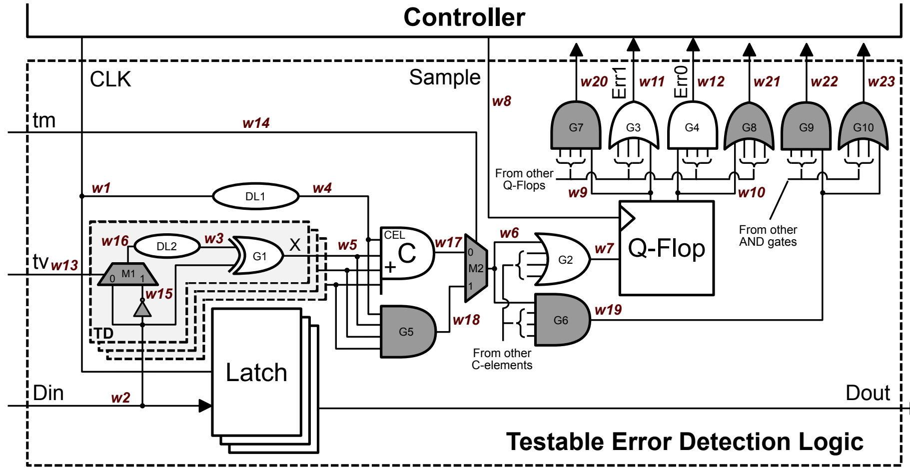

Fig. 3: TEDL diagram and labeled wires. The gray gates represent the additional logic included to improve the testability. Figure adapted from [6].

a TV, the TV is propagated to a following stage, which in turn ends up detected, and indicates that the previous stage failed to detect the injected TV. However, the data might be logically masked by the following combinational logic before it reaches the next EDS, leading to an undetected fault. These uncertainties motivated the development of a new testable EDL architecture, detailed next.

V. The Proposed Test Approach

A. Proposed Architecture

The proposed TEDL, Figure 3, is designed to add controllability and observability, for test purposes, and to avoid assumptions and uncertainties of our previous approach. As previously pointed, part of the EDL is only activated when a TV occurs. So, to properly test this part of the EDL, one must be able to control the EDL's input to generate TVs. To generate a TV, the original Blade EDS is modified. First, note that the TD is no longer incorporated into the sequential element. As previously discussed in Section IV, this opens the possibility to use scan cells that are compatible with commercial EDA tools.

The new TD has a new input called *tv*. This signal can be individually controlled for each stage or be a global input signal connected to all TDs. In this work, the global approach is applied. When the tv is activated, it forces the *X* output of all TDs to be constantly high, independently of the input *Din*, thus generating a TV. An alternative solution would be to add a two-input OR gate between each TD output and the correspondent C-element input to force a TV, but there would be less controllability and higher area overhead.

The *tm* port is another new global input signal. It controls whether the output of the C-element or the output of *G5* is forwarded. This gate and the other gray elements highlighted in Figure 3 are concurrent checkers added to detect faults that, otherwise, would be masked and not detected. For instance, a stuck-at-0 in *w5* cannot be detected if all *X* signals are at logic level 1, since the C-element output rises if at least one of its inputs plus the *CLK* signal are activated, which means that the fault will be masked and not observed.

It is important to notice that the proposed approach has little impact in Blade's timing constraints. The timing overhead of the *M1* multiplexer and the inverter inside the new TD can be compensated in the *DL2* delay line. Except for the *M2* multiplexer, all the other new elements do not create timing overheads compared to the original architecture.

The diagnostics of the internal stuck-at faults can be obtained by observing the outputs *w20*, *w11*, *w12*, *w21*, *w22* and *w23*, and the pattern associated with internal faults. These outputs are connected to scan flops to be externally observable.

B. TEDL Operational Modes

In order to stimulate the entire circuit, the TEDL must switch between its four operational modes, defined by the state of input signals *tm* and *tv*. The four modes are: normal mode (NM), normal mode with timing violation (NMTV), test mode (TM) and test mode with timing violation (TMTV). In NM, the multiplexer *M1* selects the path that does not pass through the inverter and *M2* selects the *w17* path. In NMTV, *tv* is enabled and the *w15* path is selected, which forces all *X* signals of the pipeline stage to 1. The TM is used basically to detect stuck-at faults in *w5* by selecting the *w18* path. Finally, the TMTV is applied to force all *X* signals to 1 and to pass them through the *G5* gate.

The test procedure with the TEDL requires that the pipeline is full. In Blade, this is done through an asynchronous handshaking protocol. A full pipeline will not accept a new input data request until an output data is acknowledged, and a lockstep control of the circuit is created. It allows the circuit to alternate between a test pattern capture phase and a test pattern shift phase. The full pipeline also avoids that real TVs are generated since all stages are stable. At this moment, the TEDLs can be configured to the different operation modes in order to produce the test patterns. After each capture phase, a shift phase is also executed, and the faults are identified through the fault classification method described below.

C. Fault Classification Method

A fault is detected whether the output pattern is different from the expected gold pattern defined for each operational

mode (Table II). These six output ports also enable some level of diagnostic. One important thing to notice about the fault patterns is that the state of *tm* and *tv* signals are also considered. For instance, assume that there is a stuck-at-0 at *w7*. If a TV occurs, this fault prevents the Q-Flop from registering the TV, and the fault cannot be detected in NM. Either the NMTV or the TMTV modes put all *G2* inputs at 1 and it is expected that an error is flagged for all internal lines. However, since *w7* is stuck-at-0, *w20* remains at 0, indicating the existence of a fault. Another example is a stuck-at-1 at *w19*. This wire is expected to be 1 only when *tv* is active, so in NM or TM *w22* and *w23* must be at 0. Since *w19* is stuck-at-1, *w23* rises, while *w22* remains at 0 due to other non faulty *G6* gates.

TABLE II: TEDL operation modes and their expected outputs.

Mode	Gold Pattern							
	TM	tv	w20	w11	w12	w21	w22	w23
NM	0	0	0	0	1	1	0	0
NMTV	0	1	1	1	0	0	1	1
TM	1	0	0	0	1	1	0	0
TMTV	1	1	1	1	0	0	1	1

VI. Case Study: Plasma CPU

To evaluate the fault coverage and area overhead of our proposed architecture we use the Blade automated flow described in [6]. This 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. It consists of Tcl and shell scripts, a library of custom cells and a Verilog co-simulation environment. There are 5 main steps for the circuit conversion: *Synchronous Synthesis*, *Flip-flop to Latch conversion*, *Latch Retiming*, *Resynthesis* and *Blade Conversion*. The last step was the one that was modified to include the proposed TEDL. In order to compare the results, we implemented the same case study of [6], a 3-stage version of MIPS OpenCore CPU called Plasma [11], targeting a 28nm FDSOI technology.

The co-simulation environment implements a testbench where a stream of inputs is forked to both the synchronous and Blade netlists, and the stream of outputs are compared to validate the implementation. The fault simulation environment described in [16] was incorporated into this co-simulation environment. The fault simulation environment has as input parameters: a list of fault points to inject faults, that consists of all the labeled wires presented in Figure 3; the gold patterns for each one of the operation modes, shown in Table II; the fault patterns that indicate the existence of an internal fault; and the list of faults that can be related to those fault patterns (diagnostic information).

A. Fault Coverage

A series of fault simulations were executed in order to validate the fault coverage of the TEDL, and the modifications made to the flow and co-simulation environment. Each fault point was simulated for a single stuck-at-0 and single stuck-at-1, alternating between the four test configurations in Table II to detect the injected fault. The simulation results showed that 100% of the stuck-at faults inside the TEDL are detectable, while with the original EDL, a little over 30% of the stuck-at faults are detected [16]. There are particularly three fault points inside the TEDL that depend on a transition in *Din* to produce a pattern that can be related to a fault. These are: *w15*, *w16* and *w3*. Since we are assuming no control over *Din*, the ATPG or the source code executing on the Plasma CPU must be able to generate these transitions. In addition, since these 3 nets are internal to the TD cell, they would be collapsed from the fault list generated by a conventional ATPG tool.

TABLE III: Area of EDL vs TEDL used in the Plasma CPU.

Cell	EDL		TEDL	
	N	Area μm^2	N	Area μm^2
C-element	80	189.055	80	189.055
Q-Flop	20	91.280	20	91.280
Latch	238	427.258	238	427,258
TD	238	466.099	238	893.357
OR G2	20	52.224	20	52.224
OR G3	2	7.507	2	7.507
AND G4	2	7.507	2	7.507
AND G5	-	-	80	112.934
AND G6	-	-	20	55.488
AND G7	-	-	2	7.507
OR G8	-	-	2	7.507
AND G9	-	-	2	7.507
OR G10	-	-	2	7.507
MUX M2	-	-	80	117.504
MUX-D	-	-	14	79.968
Total	362	1240.930	564	1920.606
Area Overhead				54.77%

TABLE IV: Comparison of Plasma-EDL vs Plasma-TEDL in terms of area (μm^2).

	Plasma-EDL	Plasma-TEDL	Overhead
Combinational area	7095.28324	7829.35683	9.38%
Buf/Inv area	608.89921	687.23522	11.40%
Noncombinational area	7860.69133	7860.69133	-
Macro/Black Box area	228.57500	228.57500	-
Net Interconnect area	undefined	undefined	-
Total cell area [1]	15184.54957	15918.62316	4.61%

B. Area Comparison

The area comparison of both implementations is shown in Table III and Table IV. Table III presents the EDL and TEDL number of elements and its corresponding area. Both Plasma implementations have the 238 TDs divided into two groups, each one with its own controller.

Unlike the original EDL architecture, the TD in the TEDL is not incorporated to the sequential element. To compare the original EDL architecture with the TEDL, instead of using a custom cell, the original EDS is also decomposed in

[1]Does not include Buf/Inv area and Net Interconnect area.

standard cells. The resulting area for the original TD and the new TD is $1.958\,\mu m^2$ and $3.754\,\mu m^2$ respectively. In Table III the $893.357\,\mu m^2$ TD area refers to total number of TDs ($3.754 * 238$). The test patterns are captured through a scan chain connected to the outputs *w20*, *w11*, *w12*, *w21*, *w22* and *w23* of each stage, but any observability technique can be used. For this work a MUX-D scan chain is applied and its area is acconted in Table III.

As our results show, the TEDL imposed an increment of 54.77% in area, when compared to the original EDL in the Plasma design (Table III). Note that this experiment is intended to compare the TEDL using the same scenario of Blade's proposal [6], but Blade allows a tradeoff between critical paths covered by EDLs and the area overhead of its implementation. In this particular setup, out of the total 529 latches, 238 are modified . When looking at the overall area overhead, with almost half of the laches modified, the Plasma-TEDL shows a 4.61% area overhead when compared to Plasma-EDL, and the TEDL affects mainly the combinational area of the designs (Table IV). The area can be further reduced by incorporating *M2* multiplexer and *G5* AND gate. Finally, the TD can be further optimized and some of the concurrent checkers, AND and OR gates, can be replaced by NAND and NOR gates respectively.

VII. Conclusion

This paper presents a testable error detection logic (TEDL) architecture to be used in the Blade timing resilient template. The new architecture provides the necessary infrastructure to achieve full coverage of single stuck-at faults inside the TEDL while maintaining compatibility with commercial EDA tools. By adding four modes of operation and observability of the TEDL outputs, it is possible to capture patterns that can be related to specific faults inside the TEDL. Comparing the Plasma-TEDL with the Plasma-EDL, the area overhead is 4.61%. As an ongoing work, we believe that the area overhead of the TEDL can be further reduced, and we also plan to extend the TEDL fault analysis to delay fault model.

Acknowledgment

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

References

[1] D. Ernst, N. 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 *MICRO-36*, 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 *ISSCC 2008*, Feb 2008, pp. 400–622.

[3] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. 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 JSSC*, vol. 48, no. 1, pp. 66–81, Jan 2013.

[4] M. Choudhury, V. Chandra, R. Aitken, and K. Mohanram, "Time-borrowing circuit designs and hardware prototyping for timing error resilience," *IEEE TCOMP*, vol. 63, no. 2, pp. 497–509, Feb 2014.

[5] S. Kim and M. Seok, "Variation-tolerant, ultra-low-voltage μP with a low-overhead, within-a-cycle in-situ timing-error detection and correction technique," *IEEE JSSC*, vol. 50, no. 6, pp. 1478–1490, Jun 2015.

[6] D. Hand, M. Moreira, H. Huang, D. Chen, F. Butzke, Z. Li, M. Gibiluka, M. Breuer, N. Calazans, and P. A. Beerel, "Blade - a timing violation resilient asynchronous template," in *ASYNC*, May 2015, pp. 21–28.

[7] M. Cannizzaro, S. Beer, J. Cortadella, R. Ginosar, and L. Lavagno, "Saferazor: Metastability-robust adaptive clocking in resilient circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 9, pp. 2238–2247, Sept 2015.

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

[9] R. N. Tadros, W. Hua, M. T. Moreira, N. L. V. Calazans, and P. A. Beerel, "A low-power low-area error-detecting latch for resilient architectures in 28-nm fdsoi," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 9, pp. 858–862, Sept 2016.

[10] W. Hua, R. N. Tadros, and P. A. Beerel, "Low area, low power, robust, highly sensitive error detecting latch for resilient architectures," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, 2016, pp. 16–21.

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

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

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

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

[15] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S. Lu, T. Karnik, and V. De, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE 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 *2016 22nd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, May 2016, pp. 37–42.

[17] I. Kwon, S. Kim, D. Fick, M. Kim, Y. Chen, and D. Sylvester, "Razor-Lite: A light-weight register for error detection by observing virtual supply rails," *IEEE JSSC*, vol. 49, no. 9, pp. 2054–2066, Sep 2014.

[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*, vol. 37, no. 9, pp. 1005–1018, Sep 1988.

[19] F. te Beest, A. Peeters, K. van Berkel, and H. Kerkhoff, "Synchronous full-scan for asynchronous handshake circuits," *Journal of Electronic Testing*, vol. 19, no. 4, pp. 397–406, Aug 2003.

[20] L. R. Juracy, M. T. Moreira, F. A. Kuentzer, and A. de Morais Amory, "Optimized design of an lssd scan cell," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 765–768, Feb 2017.

[21] F. te Beest and A. Peeters, "A multiplexer based test method for self-timed circuits," in *11th IEEE International Symposium on Asynchronous Circuits and Systems*, Mar 2005, pp. 166–175.

[22] H. Iwata, S. Ohtake, M. Inoue, and H. Fujiwara, "Bipartite full scan design: A dft method for asynchronous circuits," in *2010 19th IEEE Asian Test Symposium*, Dec 2010, pp. 206–211.

[23] K. Chrysanthou, P. Englezakis, A. Prodromou, A. Panteli, C. Nicopoulos, Y. Sazeides, and G. Dimitrakopoulos, "An online and real-time fault detection and localization mechanism for network-on-chip architectures," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 2, pp. 22:1–22:26, Jun. 2016.

[24] N. J. Wang and S. J. Patel, "Restore: Symptom-based soft error detection in microprocessors," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 188–201, July 2006.