Resilient Bundled-Data Design: Motivation, Results to Date, and Remaining Challenges*

Peter A. Beerel[1] and Ney L. V. Calazans[2]

[1] University of Southern California - Los Angeles, California, USA,
pabeerel@usc.edu
[2] Pontifícia Universidade Católica do Rio Grande do Sul - Porto Alegre, Brazil,
ney.calazans@pucrs.br

Abstract. The periodic nature of the global clock in traditional synchronous designs forces circuits to be margined for the worst possible case of process, voltage, temperature, and data conditions. This constrains the silicon to operate at worst-case frequencies and at conservative supply voltages. Resilient architectures promise to remove these margins, by detecting and correcting timing errors when they occur, thereby creating the potential to achieve real average-case operation. However, synchronous resilient schemes previously proposed can suffer from multiple issues, including being susceptible to metastability and requiring often complex changes to the architecture to support replay-based recovery from timing errors. These problems respectively lead to circuit failures and/or incur high timing penalties when errors occur. This paper reviews a recently proposed resilient bundled-data template called Blade that is robust to metastability issues, requires no replay-based logic, and has low timing error penalties. It also describes some open issues and new research opportunities this template presents, including automation problems to target average-case operation, specific circuit optimizations to minimize resiliency overhead, and the need for new test procedures to tune delay lines and screen out bad chips.

1 Introduction and Related Work

Traditional synchronous designs must incorporate timing guardbands to ensure correct operation under worst-case delays caused by process, voltage, and temperature (PVT) variations as well as data-dependency [6]. This is particularly important in low-power low-voltage designs, as performance uncertainty due to PVT variations grows from around 50% at nominal supply to around 2,000% in the near-threshold domain [14]. To address this problem, many synchronous design techniques for resilient circuits have been proposed that address delay variations. For example, canary FFs predict when the design is close to a setup timing failure (see e.g., [41]). Designs can then adjust their supply voltage or clock frequency either statically or dynamically to ensure correct operation at

* This is an extended and updated version of an ECCTD 2015 invited paper on the same topic.

the edge of failure. The adverse impact of variations on hold margins are significantly more challenging to manage because changing the clock frequency and voltage does not typically resolve hold problems and thus these must be very conservatively managed. Hold constraints are typically resolved by preemptively adding hold buffers to all "short" paths in the design. Unfortunately, at low voltages, the number of hold buffers needed can be much larger than at nominal voltages, because the increased delay variation causes: (1) clock uncertainty to grow; (2) a larger fraction of paths to be identified as potentially short, due to the possible decrease in delays resulting from variability; and (3) the hold buffers themselves possibly being unexpectedly fast. All these margins translate into considerable increases in energy consumption.

Several research groups have explored adding a degree of *timing resiliency* into the design to detect and then recover from setup violations [9, 12, 25, 27]. There are two general approaches: architecturally dependent, or "replay-based" approaches, and architecturally independent. The former includes Razor II [12] and the Intel approach described in [6]. The problem with these approaches is that they work much like pulsed latch circuits: the wider the pulse, the more resiliency is obtained, at the cost of worsening hold time margins [15]. Moreover they require synchronizers in the control path, incurring long delays to identify whether an error occurred, and demand complex replay and recovery mechanisms [6, 12, 27]. Granted, the area overhead of these can be amortized by reusing existing recovery logic (e.g., for resuming after a mispredicted branch), but the techniques remain architecturally invasive and thus a design challenge. In contrast, architecturally independent approaches like Bubble Razor [15] and TIMBER [9] require no architectural changes and can be automatically generated from standard RTL specifications. The flow involves replacing flip-flops with retimed latches that have non-overlapping clocks, mitigating hold time problems. Bubble Razor, for example, avoids replay and recovery by immediately stalling neighboring stages via clock gating, and solves timing errors on the fly and locally. However, the template assumes that metastability can be resolved within one clock period, which is often unrealistic and leads to poor mean-time-between-failures rates [3]. More recent work [25] proposes to borrow time only from the following stage by quickly boosting its supply voltage to accommodate for the borrowed time. Unfortunately, this approach requires fast error detection and dynamically adjustable supply voltages which limits its applicability.

Different asynchronous templates have also been proposed to address the excessive margining problem (e.g., [42]). Quasi-delay-insensitive (QDI) templates use completion signal logic which makes them robust to delay variations at the cost of increased area (often 4x larger than synchronous counterparts or even more) and high switching activity due to a return to zero paradigm (e.g., [16, 38]). Bundled-data templates (e.g., micropipelines [39]) use delay lines matched to single-rail combinational logic, providing a low area, low switching activity asynchronous solution (e.g., [10]). However, the delay lines must be implemented with sufficiently large margins in the presence of on-chip variations [11], reducing the advantages of this approach. Researchers have proposed different solutions to mitigate these margins, such as duplicating the bundled-data delay lines [8],

constraining the design to regular structures such as PLAs [24] and using soft latches [28]. Others have suggested current-based completion sensing techniques (e.g., [1, 29]) that rely on analog current sensors, which can be prohibitively power hungry.

This work focuses on a recently proposed asynchronous design template that couples the architectural benefits of resilient techniques with the flexibility of asynchronous bundled-data pipelines. The template, called Blade, minimizes hold time issues, requires no replay-based logic, and is supported by an automatic translation flow from synchronous RTL specifications. It is not only safe from metastability issues but also takes advantage of the low *average* metastability resolution times, which leads to low timing error penalties compared to synchronous alternatives. It thus provides significantly higher potential performance and voltage scaling power benefits.

The paper reviews Blade principles and operation, comparing and contrasting the approach to synchronous alternatives. Its recent application to the design of a MIPS OpenCore processor illustrates techniques to reduce overheads and maximize performance and power benefits. The paper also discusses the range of designs for which this design style is likely to provide the biggest overall benefit, as well as some of the open problems that must be solved to maximize the opportunity to use Blade and make the method commercially attractive.

2 The Blade Bundled-Data Architecture

As Figure 1 shows, pipeline stages in Blade use single-rail logic followed by Transition Detector with Time Borrowing (TDTB) error detecting latches (EDLs) [6,32], Q-Flops [35], and two reconfigurable delay lines. The stage-to-stage delay line is of duration δ and controls when the TDTB goes transparent and begins to propagate data at the output of the combinational logic to the next stage. According to the timing diagram depicted in Figure 2, the asynchronous controller speculatively assumes data at the output of the TDTB latch is stable and triggers the request to the next stage via the standard bundled data request channel consisting of $R.req$ and $R.ack$. The second delay line is of duration Δ and defines a time window during which late transitions that violate this assumption (i.e. timing errors) are allowed, which is called the *timing resiliency window* (TRW). While Δ is elapsing, CLK is high (i.e. the Data Latch is transparent).

Error detecting latches are responsible for triggering an error if a timing violation occurs during the TRW. While there are several EDL implementations (e.g., [6,9,15,32]), Blade employs a custom design [32] based on TDTB latches [6]. The basic design requirement is this component triggers an error on its E output in response to any transition or glitch during the TRW that is significant enough to also propagate to its data output [32]. In this way, no timing violation is missed.

In addition to the push data channel L, Blade uses a second pull *error channel* formed by signals $RE.req$ and $RE.ack$ to manage potential timing violations. Near the end of the TRW, after receiving a request on the $RE.req$ signal, the controller will trigger a signal that directs the Q-Flop to sample the E signal,

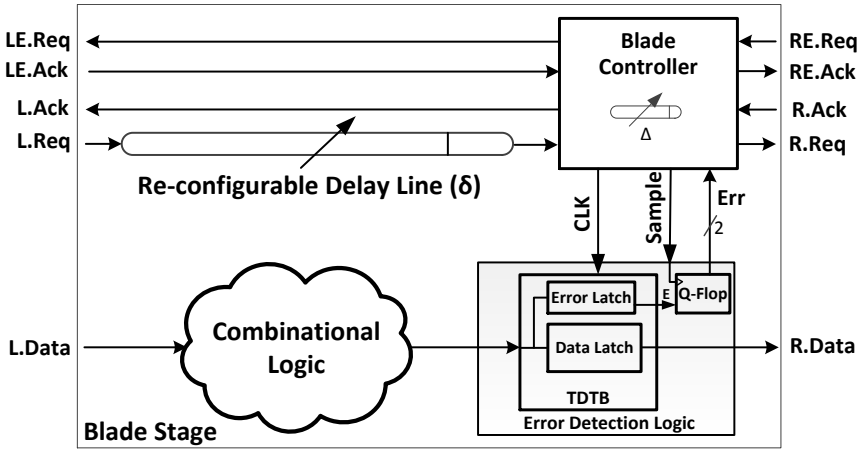

Fig. 1. The Blade architecture typical stage structure.

determining whether or not a timing error occurred during the TRW. If an error did not occur $RE.ack$ is immediately asserted, else Δ is triggered and only after that $RE.ack$ is asserted. Because the setup time of the TDTB Error Latch may be violated, the E signal may be metastable during sampling. To cope with this, the Q-Flop has a built-in metastability filter that guarantees metastability does not propagate to its Err output. In fact, this output is intentionally made a dual-rail signal that only becomes valid after the Q-Flop has safely determined if an error occurred or not. The controller simply waits for this to happen before acknowledging the error channel request via the $RE.ack$ signal. This ensures that metastability, while possibly causing an instantaneous cycle slowdown, does not propagate to the main control path. This is in stark contrast to synchronous schemes, which must wait for a fixed, larger metastability resolution time set to guarantee a sufficiently large mean time between failures (MTBF).

There are two main delay lines that affect the performance of Blade, δ and Δ. Compared to a traditional synchronous circuit, with clock period C, we set $C = \delta + \Delta$. The TRW (defined by Δ) must be large enough to capture even the worst-case datapath delay. However, a trade off in setting these values emerges, as increasing Δ allows δ to be smaller and the system to operate faster if no timing violations (errors) occur; on the other hand, the shorter stage-to-stage delay means that more transitions will occur while the latch is transparent, thereby increasing the frequency of errors that force subsequent pipeline stages to be delayed by the now larger Δ value. The optimal Δ depends greatly upon the amount of total variation (due to data and PVT variations) that can be expected in the design, and can range from 20% to over 60% [18] of the stage total delay. This is in contrast to Bubble Razor whose optimal error rate is less than 5% percent [15] and synchronous replay-based Razor schemes whose optimal error rate is less than 1% [7].

When Δ is sufficiently smaller than δ, the next stage has time to check whether the previous stage has an error before it makes its own latch transpar-

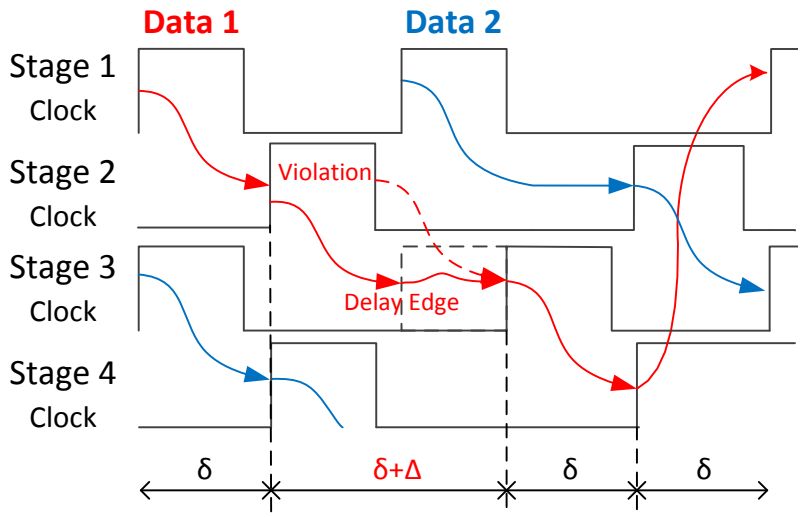

Fig. 2. Typical timing diagram for the Blade template.

ent, delaying the transparency phase if the previous stage had an error. Stage clocks will thus remain non-overlapping, as illustrated in Figure 2, making it easy to satisfy hold times. This is again in contrast to most synchronous resiliency schemes that make meeting hold time margins harder. Supporting larger values of Δ (w.r.t. δ) is also possible and is beneficial when data/process yield high variability. However, the result is that the transparency phases of neighboring stages clocks will overlap, and this may cause hold time issues similar to those seen in synchronous approaches (see [15] for an encompassing analysis). Managing these hold time issues in synchronous resiliency approaches is particularly challenging, as they cannot be fixed by slowing down the clock. Accordingly, hold times need to margined to a higher degree than setup times. As mentioned earlier, these hold margins are typically satisfied by adding hold buffers to the datapath, but the higher margins may make the number of added buffers impractically large for designs with high variability. In contrast, an asynchronous solution like Blade can easily add an additional programmable delay line to the backward control path, actively managing the degree of transparency overlap, which makes such extra margins unnecessary. In both cases the flexibility of the asynchronous solution makes managing hold time issues far more practical.

Lastly, note that Blade also uses programmable delay lines, because under significant PVT variations it may be difficult to achieve the optimal TRW, which captures the delay of all worst-case paths via static design analysis and optimization. Programmable delay lines allow customizing the actual delay post-silicon. In particular, the authors expect that during chip characterization delay lines are analyzed and optimally configured for every chip produced, subject to some quantization error. In particular, quantization errors in δ may lead to a non-optimal expected error rate, but the overall performance will remain close to

optimal [18]. Any additional margin needed to account for worst-case paths under PVT variations can be added only to the Δ delay line. Given the average frequency of timing violations can be in the range of 20%-40%, the impact of the added margin is only experienced 20-40% of the time, greatly reducing the percentage drop in performance. This is in contrast to non-resilient bundled-data designs (e.g., [10]) in which the added margin affects performance 100% of the time. As an example, a 10% increase in variation due to PVT can result in up to 30% margin penalty in synchronous designs; however, even considering a 40% rate of timing violations, the computed performance impact on Blade is less than 13% [19].

3 Preliminary CAD Flow

The authors' teams developed a preliminary flow to automatically convert single CLK domain, synchronous RTL designs to the Blade template using industry standard synthesis tools. The flow consists of various Tcl and shell scripts that drive the tools and a library of custom cells (e.g., the TDTB error latch), needed to make the template efficient.

In addition, to further reduce area and power overheads of the error detection logic, two microarchitectural optimizations are used. First, not every pipeline stage need be error-detecting, and non error-detecting stages can time borrow. Time-borrowing stages permit data to pass through the latch during the entire time it is transparent without flagging violations. The authors found that alternating between error-detecting and time-borrowing stages can work well as this effectively halves the overhead of error detection logic while still providing sufficient resiliency. Secondly, only latches that terminate near-critical paths [19] need to be error detecting, further reducing the number of EDLs in the entire design.

As Figure 3 illustrates, the flow has five main steps:

1) **Synchronous Synthesis:** The synchronous RTL is synthesized to a flip-flop (FF-based) design for given clock.
2) **FF to Latch Conversion:** FFs are converted to master-slave latches by synthesizing the design using a fake library of standardized D flip-flops (DFFs) that can be easily mapped to standard cell latches.
3) **Latch Retiming:** The latch-based netlist is retimed using a target TRW, where the combined path delay constraint of any two stages equals the given clock period. The purpose is to split the critical path in two parts, which enables hiding inter-stage Blade handshaking overheads.
4) **Resynthesis:** The retimed netlist is then resynthesized to reduce the number of TDTBs and increase performance of the final resilient netlist. In particular, re-synthesizing the logic happens such that the delay to a subset of latches is sufficiently fast to guarantee that data is stable before the latches go transparent (i.e., is not near-critical). This means that the latches do not need to be error-detecting, reducing the EDL overhead, and potentially reduces the error rate at the expense of increasing the datapath logic area.

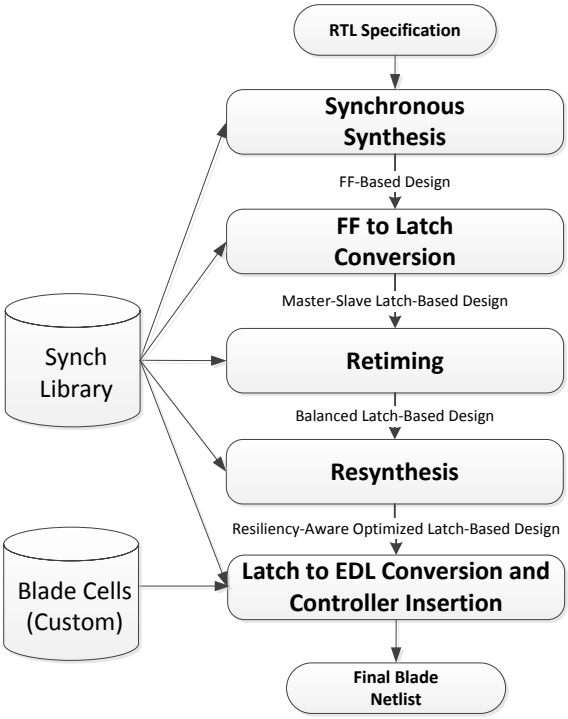

Fig. 3. The Blade design flow.

Targeting latches that cause the most errors in typical applications can lead to significant reductions in error rates with marginal increase in area. In [19] the authors employ a simple brute-force search, but more powerful means of identifying which subset of latches to speed up is an interesting area of future work.

5) **Blade Conversion:** The resynthesized latch-based netlist is then converted to the Blade template, by removing clock trees and replacing these with Blade controllers. The control logic, delay lines, and error detection logic are also inserted to create a final Blade netlist. There are many ways to implement the control logic [5]; using burst-mode specifications was explored in [19].

The authors' preliminary pre-P&R flow was tested and evaluated on a 3-stage version of Plasma [33], a MIPS OpenCore CPU, targeting a 28nm FD-SOI technology. The gate-level Blade design was compared to the equivalent synchronous design, and post-synthesis results demonstrate that for an area overhead of 8.4%, the Blade version of Plasma achieves a 19% average performance boost with a timing resiliency window of 30%. Out of the 8.4% area overhead, 32% is due to the use of EDLs and to the FF to latch conversion. With the removal of synchronous PVT margins, it led to an estimated 30%-40% improvement in performance [19].

4 Recent Developments and Open Research Problems

The technology is being commercialized by Reduced Energy Microsystems [34], a semi-conductor start-up company co-founded by William Koven, Dylan Hand, and Eleazar Vega-Gonzalez. They recently designed and fabricated a Blade-based design for light-weight encryption cyphers [20] and have plans on using Blade to build energy-efficient processors for the Internet of Things market. The success of the commercialization of Blade, however, will likely require solving several research challenges which we outline here.

4.1 Scope and Scale of Design

REM's recent work involved extending the flow illustrated in Figure 3 to start from a synthesizable-subset of SystemVerilogCSP [36] and include back-end place-and-route. In particular, they leveraged the USC-developed tool CSP2RTL to automatically decompose CSP designs into blocks of synthesized unconditional logic surrounded by efficient SEND/RECEIVE primitives. This tool allows asynchronous designers to couple the benefits of hierarchical decomposition and conditional communication with resilient pipelined designs and is based on a similar framework used for industrial-scale QDI designs [4].

Interestingly, the work in [20] represents just the beginning of what is possible when the scope of synthesizable CSP specifications is expanded. For example, we envision supporting arbitration in CSP where the CSP2RTL tool would automatically insert arbitrated merge blocks to enable the automatic design of complex routers and NoC designs. Moreover, we believe we can support mixed-timing interfaces in SystemVerilogCSP for which the tool will automatically insert clock-domain-crossing circuits. This will enable seamless integration of resilient bundled-data blocks within otherwise synchronous designs.

4.2 Area and Energy Efficiency

A CSP-based resilient design flow can support energy-efficient hardware design for a large variety of applications, but the added complexity associated with hierarchical design and conditional communication must be managed. In particular, over-decomposition can lead to increased overheads in terms of extra pipeline stages, excess error-detecting logic, and more delay lines and control logic. To properly navigate this increased design space, high-level energy and power estimation tools that guide decomposition will be important. In addition, the use of slack-less controllers such as in [20] can often provide much of the benefits of conditionality without the area overhead of a distinct pipeline stage. This design exploration should be guided by a notion of average-performance and average-energy consumption that considers the probabilities implicit in the conditional communication and the probabilities associated with error-rates at each error-detecting pipeline stage. In particular, understanding the implication of clustering latches into pipeline stages will likely be critical for a comprehensive flow that can accommodate industry-scale applications. Too few pipeline

stages and the design would loose the benefits of average-case data whereas too many pipeline stages would lead to too high overhead. Finding the right balance is likely design and use-case specific and will demand new tools to guide the design.

Even for simple RTL-based resilient bundled-data designs, numerous advances in area and power efficiency will likely be essential to the success of the technology. In particular, the design of the delay line and error-detecting logic is critical to efficiency. We expect one of the significant advantages of this design style is its ability to naturally support dynamic voltage scaling [21, 40]. Unlike a synchronous design in which adjusting voltages often requires stalling the pipeline for many clock cycles while the clock source is reconfigured, bundled-data circuits can be designed to automatically adapt to voltage scaling. If the delay line tracks the delay of the combinational logic, it need not be re-programmed when the supply voltage is changed. To support this strategy, we recently proposed a framework for delay line design [40] in which we minimize average energy subject to two-sided voltage scaling constraints. In fact, we anticipate we will need to build a library of delay elements and a CAD tool that chooses which delay elements to use based on an analysis of the likely critical path of a pipeline stage and its voltage scaling properties.

Minimizing the cost of error-detecting logic latches is also important. There are two distinct approaches to this problem. The first approach is re-synthesis in which new constraints are added to the logic synthesis / physical design to make certain latches non-critical, thereby saving the overhead of making them error-detecting. In [23], we developed a geometric programming based mathematical algorithm that guides re-synthesis to minimize the total area of the design. We have found that this often reduces error rates, but explicitly modeling and considering average-case performance is interesting future work. Moreover, we are exploring resilient-aware, latch-based retiming. Recall that the Blade CAD flow described above involves replacing FFs with a pair of latches and retiming of the slave latches to create a balanced latch based design. This balance aids in hiding the performance overhead of the asynchronous control and mitigating hold-time problems. Commercial tools support retiming of sequential elements, including latches, but the results are often sub-optimal for resilient designs as their retiming algorithm does not understand the inherent trade-off associated with near-critical paths and the error-detecting/non-error-detecting latch at which the path ends. The second approach is to design efficient multi-bit error-detecting latches. Amortizing the cost of memorizing whether an error occurs can lead to significant benefits in terms of area and power [22].

While most of the circuit design research has focused on super and near-threshold design, another important domain for Blade designs may be sub-threshold operation, particularly for the sub-set of the market in which performance is not important. Sub-threshold design, however, introduces new challenges in guaranteeing reasonable static noise margins and minimizing leakage currents. Fortunately, techniques to achieve efficient sub-threshold designs for synchronous circuits are well-known [2]. Using these techniques to design efficient efficient asynchronous control circuits, delay lines, and error-detecting

latches for the sub-threshold operation is an interesting and important area of research.

Finally, numerous researchers have developed bundled-data design flows using commercially-supported physical design tools [11, 16, 17] and REM has developed a prototype flow for Blade circuits [20]. To extend these to complex Blade designs, however, a few more enhancements will be necessary. First, new standard-cells must be designed, including efficient error-detecting latches and mutual exclusion elements. In addition, supporting the non-standard timing constraints and trade-offs associated with the introduction of programmable delay lines in complex Blade designs is novel and challenging. A naïve implementation can lead to a quadratic explosion of delay lines between interacting Blade stages. Instead, an intelligent sharing of delay lines is needed to guarantee using only a linear number of delay lines and this sharing should be guided by a variety of factors.

4.3 Design for Test, Debug, and Manufacturability

Traditional synchronous testing methodologies are based on an implicit assumption of statically controlled voltages and clocks and that the associated control logic is minimal (an on-chip PLL and off-chip voltage regulator). Traditional test methodologies have thus focused on the max-delay constraints in the core digital logic and relied on functional tests to cover the control logic. However, bundled-data resilient designs are more complex as they have programmable delay lines in every pipeline stage and have error-detecting logic that indicates when setup failures occur. One recent study explored the testability of the Blade template and found while many faults were implicitly testable by the error-detecting logic, other faults led to excessive errors or disabled the error-detecting capability of the circuit [26]. The complex nature of testing these circuits warrants the study of a holistic test methodology that encompasses new resiliency-aware fault models, test coverage, test generation, and design for test. This will include test methods for optimally tuning the programmable delay lines based perhaps on *in situ* error rate monitoring, as well as means to identify and discard chips with delay variations too large to correct.

5 Discussion and Conclusions

Asynchronous design has become an increasingly attractive alternative to synchronous design in several applications for a variety of reasons. For example, Intel showed that high-performance quasi-delay-insensitive (QDI) design is sufficiently robust and effective for high performance networking chips [13]. Moreover, the challenges of managing a global clock in large neuromorphic chips, have driven IBM [31] and Stanford [30] to adopt an asynchronous mostly QDI interconnect. Other academic researchers have found that built-in flow-control in bundled-data network-on-chips lead to significant benefits in terms of latency and area compared to synchronous counterparts [16]. However, efforts to commercialize bundled-data pipelines for processors demonstrated only marginal performance

benefits [10]. We hope that adding resiliency opens the door for much larger performance advantages to a broader range of applications.

Generally speaking, the range of architectures and applications for which resiliency adds value depends on two factors: the overhead one can expect from the error-detecting latches and the variance of the data and PVT dependent delays [37]. The benefits of a resilient design are higher when the fraction of combinational to sequential area is large, because the relative overheads of the TDTBs is smaller. Thus, resilient design favors less pipelined designs. Moreover, an architecture where the difference between average and worst-case delay is large will likely benefit more than a well balanced architecture and even more likely if the worst-case paths are rarely executed [37]. For example, architectures that involve complex logic with rarely executed long carry chains will benefit more than balanced designs consisting of many regular structures (e.g., memories). Fortunately, there are many architectural decisions that can be made to favor timing resilient templates [37].

Lastly, it is important to emphasize that the advantages of asynchronous resilient designs are difficult to approximate in synchronous architectures. In particular, asynchronous resilient designs adapt to the quite low average-case time it takes for metastability to resolve, which in principle can be unbounded. In contrast, the periodic nature of the clock forces synchronous alternatives to be designed for a much larger fixed resolution time, set by an acceptable MTBF. This difference enables our solution to be architecturally-independent, whereas existing robust synchronous solutions are forced to be based on recover and replay logic to obtain reasonable MTBF.

Thus, we believe asynchronous resiliency is a promising research direction to obtain efficient designs which adapt to the combination of PVT and data variations and naturally supports voltage scaling. We believe that a good initial market for this technology is the Internet of Things market, but the higher energy efficiency may very well be attractive to more general computing domains.

References

1. Akgun, O., Leblebici, Y., Vittoz, E.: Current Sensing Completion Detection for Subthreshold Asynchronous Circuits. In: European Conference on Circuit Theory and Design (ECCTD). pp. 376–379 (Aug 2007)
2. Alioto, M.: Ultra-low power VLSI circuit design demystified and explained: A tutorial. IEEE Transactions on Circuits and Systems - I: Regular Papers 59(1), 3–29 (Jan 2012)
3. Beer, S., Cannizzaro, M., Cortadella, J., Ginosar, R., Lavagno, L.: Metastability in Better-Than-Worst-Case Designs. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 101–102 (Apr 2014), (Fresh Ideas Workshop)
4. Beerel, P.A., Dimou, G.D., Lines, A.M.: Proteus: An ASIC flow for GHz asynchronous designs. IEEE Design & Test of Computers 28(5), 36–51 (2011)
5. Beerel, P., Ozdag, R., Ferreti, M.: A Designer's Guide to Asynchronous VLSI. Cambridge University Press (2010)

6. Bowman, K., Tschanz, J., Kim, N., Lee, J., Wilkerson, C., Lu, S., Karnik, T., De, V.: Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance. IEEE Journal of Solid State Circuits 44(1), 49–63 (Jan 2009)
7. Bowman, K., Tschanz, J., Lu, S., Aseron, P., Khellah, M., Raychowdhury, A., Geuskens, B., Tokunaga, C., Wilkerson, C., Karnik, T., De, V.: A 45 nm resilient microprocessor core for dynamic variation tolerance. IEEE JSSC 46(1), 194–208 (Jan 2011)
8. Chang, I.J., Park, S.P., Roy, K.: Exploring Asynchronous Design Techniques for Process-Tolerant and Energy-Efficient Subthreshold Operation. IEEE Journal of Solid State Circuits 45(2), 401–410 (Feb 2010)
9. Choudhury, M., Chandra, V., Aitken, R., Mohanram, K.: Time-Borrowing Circuit Designs and Hardware Prototyping for Timing Error Resilience. IEEE Transactions on Computers 63(2), 497–509 (Feb 2014)
10. Cortadella, J., Kondratyev, A., Lavagno, L., Sotiriou, C.: Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(10), 1904–1921 (Oct 2006)
11. Cortadella, J., Lupon, M., Moreno, A., Roca, A., Sapatnekar, S.: Ring Oscillator Clocks and Margins. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 19–26 (May 2016)
12. Das, S., Tokunaga, C., Pant, S., Wei-Hsiang, M., Kalaiselvan, S., Lai, K., Bull, D., Blaauw, D.: Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance. IEEE Journal of Solid State Circuits 44(1), 32–48 (Jan 2009)
13. Davies, M., Lines, A., Dama, J., Gravel, A., Southworth, R., Dimou, G., Beerel, P.: A 72-Port 10G Ethernet Switch/Router using Quasi-Delay-Insensitive Asynchronous Design. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 103–104 (May 2014), (Industrial Track)
14. Dreslinski, R., Wieckowski, M., Blaauw, D., Sylvester, D., Mudge, T.: Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. Proceedings of the IEEE 98(2), 253–266 (Feb 2010)
15. Fojtik, M., Fick, D., Kim, Y., Pinckney, N., Harris, D., Blaauw, D., Sylvester, D.: 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 48(1), 66–81 (Jan 2013)
16. Ghiribaldi, A., Bertozzi, D., Nowick, S.: A Transition-signaling Bundled Data NoC Switch Architecture for Cost-effective GALS Multicore Systems. In: Design & Test in Europe (DATE). pp. 332–337 (Mar 2013)
17. Gibiluka, M., Moreira, M.T., Calazans, N.L.V.: A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework. In: Euromicro Conference on Digital System Design (DSD). pp. 79–86 (2015)
18. Hand, D., Huang, H., Cheng, B., Zhang, Y., Moreira, M.T., Breuer, M., Calazans, N.L.V., Beerel, P.A.: Performance Optimization and Analysis of Blade Designs under Delay Variability. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 61–68 (May 2015)
19. Hand, D., Moreira, M., Huang, H., Chen, D., Butzke, F., Li, Z., Gibiluka, M., M., B., Calazans, N.L.V., Beerel, P.A.: Blade - A Timing Violation Resilient Asynchronous Template. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 21–28 (May 2015)
20. Hand, D., Katzin, A., Koven, W.: Adding Conditionality to Resilient Bundled-Data Designs. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC) (May 2016), (Industrial Track)

21. Heck, G., Heck, L.S., Singhvi, A., Moreira, M.T., Beerel, P.A., Calazans, N.L.V.: Analysis and Optimization of Programmable Delay Elements for 2-Phase Bundled-Data Circuits. In: International Conference on VLSI Design (VLSI). pp. 321–326 (Jan 2015)
22. Hua, W., Tadros, R., Beerel, P.: Low area, low power, robust, highly sensitive error detecting latch for resilient architectures. In: International Symposium on Low Power Electronics and Design (ISLPED) (Aug 2016)
23. Huang, H.H.H., Cheng, H., Chu, C.C., Beerel, P.A.: Area optimization of resilient designs guided by a mixed integer geometric program. In: Design Automation Conference (DAC) (Jun 2016)
24. Jayakuma, N., Garg, R., Gamache, B., Khatri, S.: A PLA based Asynchronous Micropipelining Approach for Subthreshold Circuit Design. In: Design Automation Conference (DAC). pp. 419–424 (Jun 2006)
25. Kim, S., Seok, M.: Variation-Tolerant, Ultra-Low-Voltage μP With a Low-Overhead, Within-a-Cycle In-Situ Timing-Error Detection and Correction Technique. IEEE Journal of Solid State Circuits 50(6), 1478–1490 (Jun 2015)
26. Kuentzer, F., Amory, A.: Fault classification of the error detection logic in the blade resilient templates. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 37–42 (May 2016)
27. Kulkarni, J.P., Tokunaga, C., Aseron, P.A., Nguyen, T., Augustine, C., Tschanz, J.W., De, V.: A 409 GOPS/W Adaptive and Resilient Domino Register File in 22 nm Tri-Gate CMOS Featuring In-Situ Timing Margin and Error Detection for Tolerance to Within-Die Variation, Voltage Droop, Temperature and Aging. IEEE Journal of Solid State Circuits 51(1), 117–129 (Jan 2016)
28. Liu, J., Nowick, S., Seok, M.: Soft MOUSETRAP: A Bundled-Data Asynchronous Pipeline Scheme Tolerant to Random Variations at Ultra-Low Supply Voltages. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 1–7 (May 2013)
29. Liu, T.T., Alarcon, L., Pierson, M., Rabaey, J.: Asynchronous Computing in Sense Amplifier-Based Pass Transistor Logic. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC). pp. 105–115 (Apr 2008)
30. Merolla, P., Arthur, J., Alvarez, R., Bussat, J.M., Boahen, K.: A Multicast Tree Router for Multichip Neuromorphic Systems. IEEE Transactions on Circuits and Systems - I: Regular Papers 61(3), 820–833 (Mar 2014)
31. Merolla, P., et al.: A Million Spiking-neuron Integrated Circuit with a Scalable Communication Network and Interface. Science 345(6197), 668–673 (2014)
32. Moreira, M.T., Hand, D., Calazans, N.L.V., Beerel, P.A.: TDTB Error Detecting Latches: Timing Violation Sensitivity Analysis and Optimization. In: International Symposium on Quality Electronic Design (ISQED). pp. 379–383 (Mar 2015)
33. Plasma CPU, 2014. Available: http://opencores.org/project,plasma
34. Reduced Energy Microsystems. http://www.remicro.com, Accessed: March, 2016
35. Rosenberger, F., Molnar, C., Chaney, T., Fang, T.P.: Q-modules: Internally Clocked Delay-insensitive Modules. IEEE Transactions on Computers 37(9), 1005–1018 (Sep 1988)
36. Saifhashemi, A., Beerel, P.A.: SystemVerilogCSP: Modeling digital asynchronous circuits using systemverilog interfaces. In: Proceedings of Communicating Process Architectures - WoTUG-33 (Jun 2011)
37. Sartori, J., Kumar, R.: Exploiting Timing Error Resilience in Processor Architecture. ACM Transactions on Embedded Computing Systems 12(2), 89:1–89:25 (May 2013)

38. Stevens, K.S., Gebhardt, D., You, J., Xu, Y., Vij, V., Das, S., Desai, K.: The Future of Formal Methods and GALS Design. Electronic Notes in Theoretical Computer Science 245(0), 115–134 (2009)
39. Sutherland, I.E.: Micropipelines. Communications of the ACM 32(6), 720–738 (Jun 1989)
40. Tadros, R., Hua, W., Gibiluka, M., Moreira, M.T., Calazans, N.L.V., Beerel, P.A.: Analysis and design of delay lines for dynamic voltage scaling applications. In: IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC) (May 2016)
41. Tschanz, J., Bowman, K., Walstra, S., Agostinelli, M., Karnik, T., De, V.: Tunable Replica Circuits and Adaptive Voltage-Frequency Techniques for Dynamic Voltage, Temperature, and Aging Variation Tolerance. In: Symposium on VLSI Circuits. pp. 112–113 (2009)
42. Yakovlev, A., Vivet, P., Renaudin, M.: Advances in Asynchronous Logic: From Principles to GALS & NoC, Recent Industry Applications, and Commercial CAD Tools. In: Design & Test in Europe (DATE). pp. 1715–1724 (Mar 2013)