28.6 Bubble Razor: An Architecture-Independent Approach to Timing-Error Detection and Correction

Matthew Fojtik[1], David Fick[1], Yejoong Kim[1], Nathaniel Pinckney[1], David Harris[2], David Blaauw[1], Dennis Sylvester[1]

[1]University of Michigan, Ann Arbor, MI
[2]Harvey Mudd College, Claremont, CA

Several methods that eliminate timing margins by detecting and correcting transient delay errors have been proposed [1-5]. These Razor-style systems replace critical flip-flops with ones that detect late arriving signals, and use architectural replay to correct errors. However, none of these methods have been applied to a complete commercial processor due to their architectural invasiveness. In addition, these Razor techniques introduce significant hold time constraints that are difficult to meet given worsening timing variability. To address these two issues we propose Bubble Razor (B-Razor) (Fig. 28.6.1), which uses a novel error-detection technique based on two-phase latch timing and a local replay mechanism that can be inserted automatically in any design. The error detection technique breaks the dependency between minimum delay and speculation window, restoring hold-time constraints to conventional values and allowing timing speculation of up to 100% of nominal delay. The large timing speculation makes Bubble Razor especially applicable to low-voltage designs where timing variation grows exponentially.

We implemented B-Razor in an ARM Cortex-M3 microprocessor without detailed knowledge of its internal architecture to demonstrate its automated nature. The flip-flop-based design was converted to two-phase latch timing using commercial retiming tools. B-Razor was then inserted using automatic scripts. This system is the first implementation of a Razor-style scheme on a complete, commercial processor, and it provides an energy-efficiency improvement of 60% or a throughput gain of 100%.

During normal, error-free operation, data arrives at a latch input before the latch opens and no time borrowing occurs. If data arrives after the latch opens due to running at the edge of failure, B-Razor flags an error. The key observation is that these errors do not immediately corrupt processor state as they borrow time from later pipeline stages. A failure will occur when data arrives after the latch closes, which can arise if the time borrowing effect is not corrected and compounds through multiple stages.

Upon detection of a timing error, it is critical to recover quickly before time borrowing accumulates to a point of failure. Error clock gating control signals (bubbles) are propagated to neighboring latches (Fig. 28.6.1). A bubble causes a latch to skip its next transparent clock phase, giving it an additional cycle for correct data to arrive. Since it is not possible to cause all latches in the design to inject a bubble in one cycle, bubbles are propagated with each cycle from neighbor to neighbor in a wave-like pattern. A key challenge lies in how to prevent bubbles from propagating indefinitely along loops and forwarding paths and bring the circuit back to a consistent, bubble-free state. To address this, we propose a novel bubble propagation algorithm: (1) a latch that receives a bubble from one or more of its neighbors, stalls and sends its other neighbors (input and output) a bubble one half-cycle later; (2) a latch that receives a bubble from all of its neighbors stalls but does not send out any bubbles (Fig. 28.6.2) Despite the fact that latches stall at different times, the system provably maintains correct operation with every latch in the design stalling exactly once. The stalling technique is agnostic to state machine architecture or structure, allowing bubble clock gates and control logic to be automatically inserted. The only change to the external behavior of the system is an occasional single stall cycle on the inputs and outputs.

Flip-flops in the M3 were split into latches and the design was retimed using automatic tools. Retiming can be performed to the same timing constraint with 21% area overhead as the two latches per original flip-flop become an average of 3.29 latches after retiming. Logic delay in each phase was balanced such that no time borrowing occurs during error-free operation.

To reduce bubble propagation logic overhead, latches that share neighbors were automatically grouped together into clusters. A positive and negative graph was extracted based on latch connectivity and clusters were assigned using a hypergraph partitioning tool [6]. Latches in each cluster share a gated clock and combine their error signals into a common cluster error signal. A cluster takes in error signals from its fan-in clusters and sends bubbles to its neighboring clusters, causing them to stall.

Error detection circuits (Fig. 28.6.3) use a shadow latch based on [5] and wide dynamic ORs to combine error signals and bubbles, followed by latches to hold their results. As with other Razor systems, the speculation window can be limited by either the technique or the amount of latches with error checking. As this design is a demonstration vehicle, timing error checking was added to all latches to find the maximum speculation window: one clock phase minus the propagation delay of the error detection circuits, which provides ~55% timing speculation in this design. When combined with retiming overhead, this led to an artificially large cell area overhead of 87%. By only checking critical latches and limiting the speculation window size, fewer latches will require transition detection. Only 8% of latches required checking in [5]. Additionally, more efficient error checking is possible using transition detectors based on [3]. Additional B-Razor area overhead from the cluster control logic is 16%, which can also be reduced by monitoring a subset of latches.

Although the design uses dynamic cells and latch-based timing, the models given to synthesis, placement, and routing software are fully static and edge-based. Since the dynamic ORs are always followed by more ORs or a latch, the ORs are modeled as static and the latch is modeled as a flip-flop. Latches in the datapath are modeled as flip-flops, since time borrowing during error-free operation is disallowed. The resulting design appears to the tool chain as a standard, flip-flop based design with clock gating, allowing fully automated, standard integration with no designer intervention.

To interface B-Razor M3 with SRAM, wrapper logic was placed around SRAMs to make them appear as level-sensitive latches (Fig. 28.6.4). To avoid writing incorrect data to SRAM, the system uses a commercial two-port, high-speed SRAM that separates read and write ports. Writes are clocked on the negative edge of the clock when data is guaranteed to be error free. A single entry store buffer could alternately be used to stabilize writes. Since reads cannot be delayed without reducing system performance, they continue speculatively at the positive edge. Upon receiving a bubble, the memory uses the available cycle to repeat the read with the correct inputs that are captured by a bank of flip-flops on the negative clock edge. These approaches to handling SRAM in B-Razor are not specific to the M3 and can be automatically added to any system.

At 85°C with 10% supply drop, 2σ process variation, and 5% safety margin, the maximum operating frequency of the M3 design is measured as 200MHz, setting a frequency ceiling for a conventional design. With B-Razor the design can be tuned to the point of first failure (PoFF) which was 290/333/363 MHz for three shown chips, increasing throughput by 45, 67, and 82% (Fig. 28.6.5). Alternatively, supply voltage can be lowered at iso-performance, reducing M3 energy consumption by 43, 54, and 60%, respectively.

Figure 28.6.6 shows system behavior when sweeping frequency or voltage beyond the PoFF. As frequency increases, throughput improvement slows down and eventually reverses due to stall cycles. Similarly, power is reduced until timing errors become too common. Overall, an additional 22% performance or 17% energy reduction is obtained from running beyond the PoFF. This is significantly better than previous Razor approaches since only a single cycle is lost per corrected error.

References:

[1] D. Bull, et al., "A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation," *ISSCC Dig. Tech. Papers*, pp. 284-285, 2010.
[2] J. Tschanz, et al., "A 45nm resilient and adaptive microprocessor core for dynamic variation tolerance," *ISSCC Dig. Tech Papers,* pp. 282-283, 2010.
[3] D. Blaauw, et al., "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance," *ISSCC Dig. Tech. Papers,* pp. 400-401, 2008.
[4] K. Bowman, et al., "Energy-Efficient and Metastability-Immune Timing-Error Detection and Instruction-Replay-Based Recovery Circuits for Dynamic-Variation Tolerance," *ISSCC Dig. Tech Papers*, pp. 402-403, 2008.
[5] D. Ernst, et al., "Razor: a low-power pipeline based on circuit-level timing speculation," *IEEE International Symp. on Microarchitecture*, pp. 7-18, 2003.
[6] G. Karypis, et al., "Multilevel hypergraph partitioning: applications in VLSI domain," *IEEE Design Automation Conf.*, pp. 526-529, 1997.

 978-1-4673-0377-4/12/$31.00

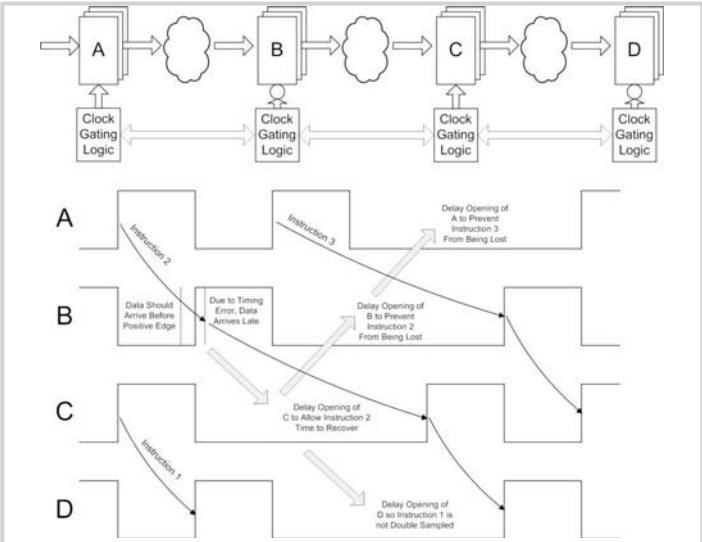


Figure 28.6.1: Timing errors are corrected by sending bubbles downstream, which must be further propagated through the circuit.

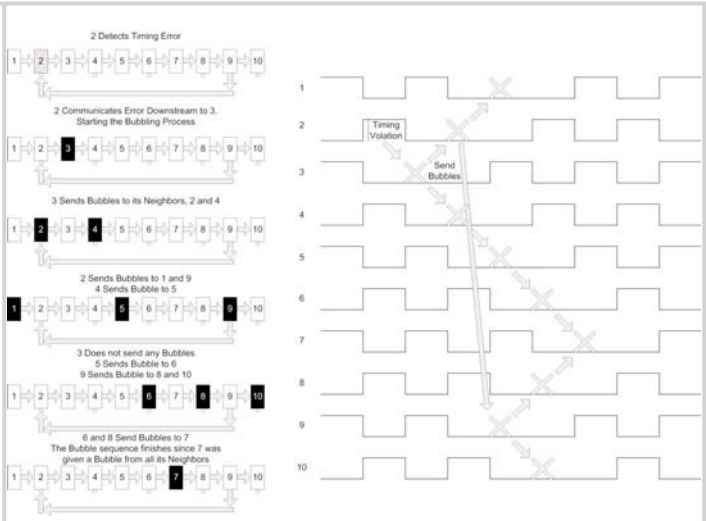


Figure 28.6.2: A latch stalls when sent a bubble by one or more neighbors, and propagates the bubble to its other neighbors.

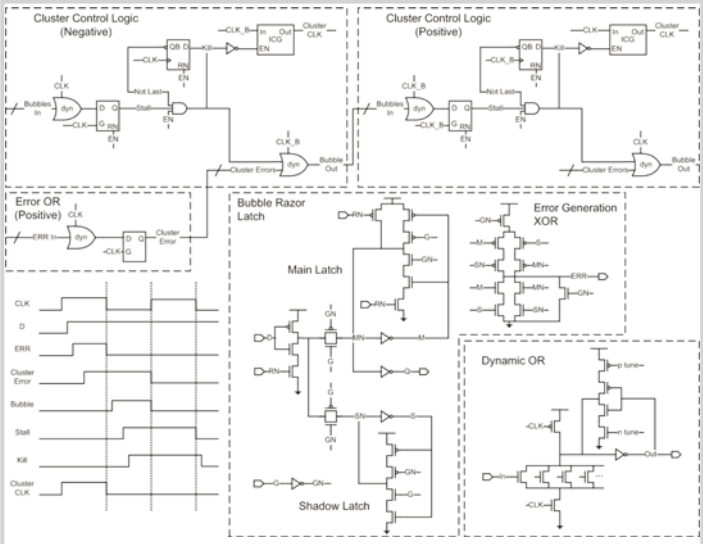


Figure 28.6.3: Bubbles are combined using dynamic OR gates. A cluster ignores bubbles if it stalled in the previous cycle.

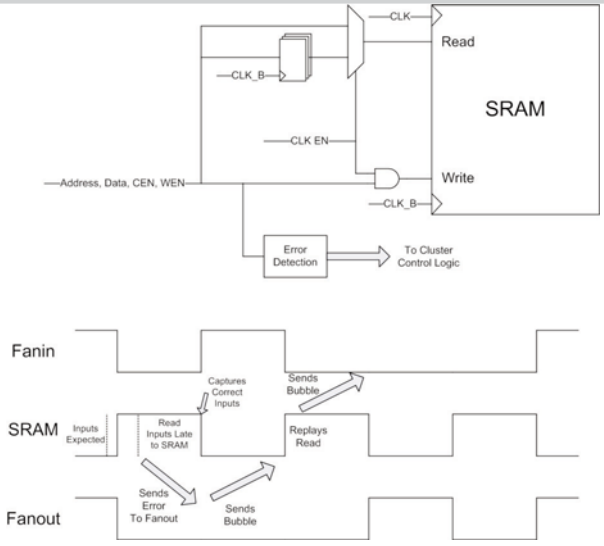


Figure 28.6.4: Wrapper logic is placed around the SRAM to achieve correct operation during timing errors and bubbles.

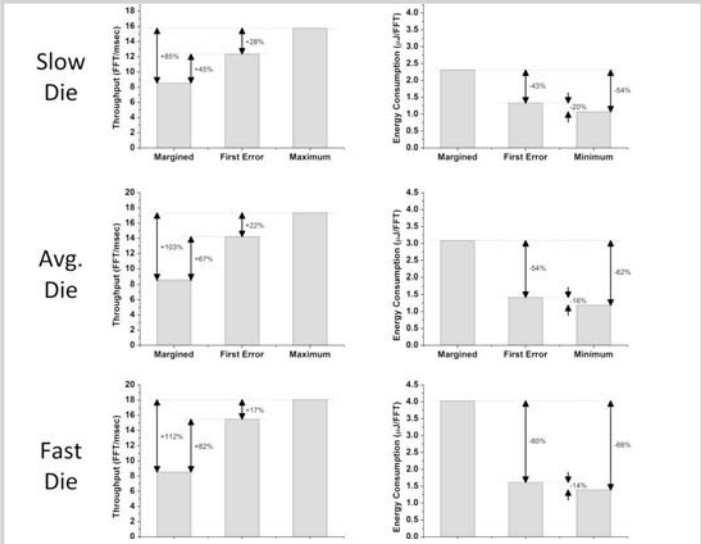


Figure 28.6.5: By running at or beyond the PoFF instead of with worst case margins, performance or energy can be improved.

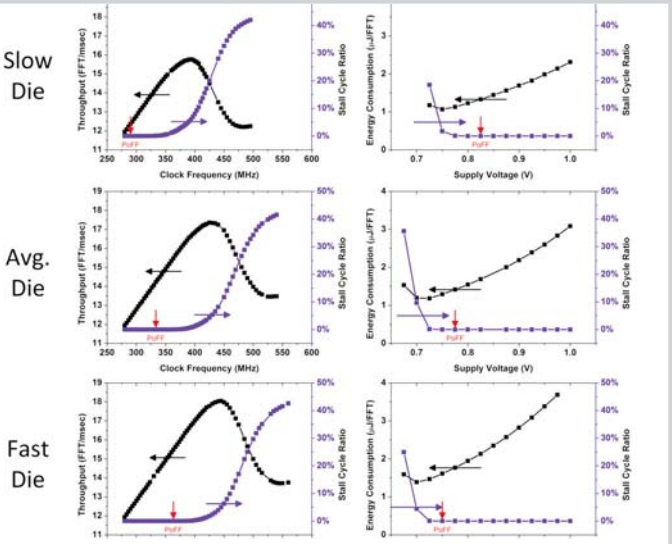
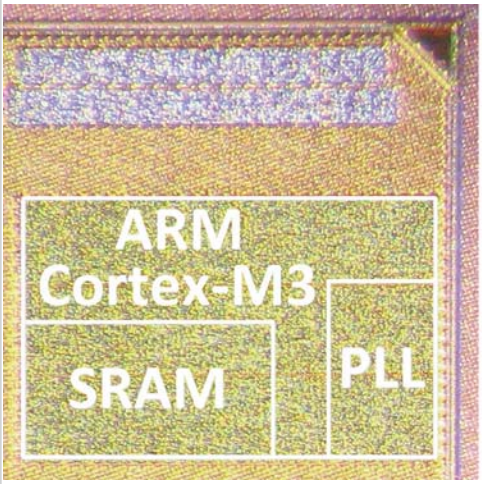


Figure 28.6.6: Due to only a single cycle penalty for fixing timing errors, gains can be made by running beyond the PoFF.



Processor Core	ARM Cortex-M3
Process Technology	IBM 45nm SOI12S0
Nominal VDD	1.0 V
SRAM Size	16 kB
Latches	7159
Positive Clusters	70
Negative Clusters	100
Speculation Window	55%

Figure 28.6.7: Die Shot and System Information.

978-1-4673-0377-4/12/$31.00