

Making Typical Silicon Matter with Razor



A codesign methodology incorporates timing speculation into a low-power microprocessor pipeline and shaves energy levels far below the point permitted by worst-case computation paths.

Todd Austin
David Blaauw
Trevor Mudge
University of Michigan

Krisztián Flautner
ARM Ltd.

An old adage says, “If you’re not failing some of the time, you’re not trying hard enough.” To address the power challenges that current on-chip densities pose, we adapted this precept to circuit design. Razor,¹ a voltage-scaling technology based on dynamic detection and correction of circuit timing errors, permits design optimizations that tune the energy in a microprocessor pipeline to typical circuit operational levels. This eliminates the voltage margins that traditional worst-case design methodologies require and allows digital systems to run correctly and robustly at the edge of minimum power consumption. Occasional heavyweight computations may fail and require additional time and energy for recovery, but the overall computation in the optimized pipeline requires significantly less energy than traditional designs.

Razor supports timing speculation through a combination of architectural and circuit techniques, which we have implemented in a prototype Razor pipeline in 0.18-micrometer technology. Simulation results of the SPEC2000 benchmarks showed energy savings for every benchmark, up to a 64 percent savings with less than 3 percent performance impact for error recovery.

SPEED, ENERGY, AND VOLTAGE SCALING

Both circuit speed and energy dissipation depend on voltage.

The speed or clock frequency, f , of a digital circuit is proportional to the supply voltage, V_{dd} :

$$f \propto V_{dd}$$

The energy E necessary to operate a digital circuit for a time duration T is the sum of two energy components:

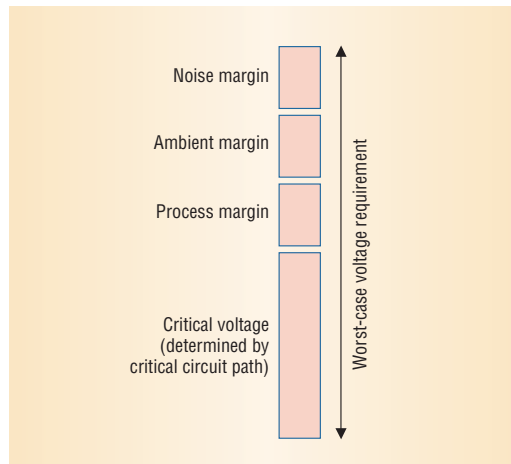
$$E = SCV_{dd}^2 + V_{dd}I_{leak}T$$

where the first term models the *dynamic power* lost from charging and discharging the capacitive loads within the circuit and the second term models the *static power* lost in passive leakage current—that is, the small amount of current that leaks through transistors even when they are turned off. The dynamic power loss depends on the total number of signal transitions, S , the total capacitance load of the circuit wire and gates, C , and the square of the supply voltage. The static power loss depends on the supply voltage, the rate of current leakage through the circuit, I_{leak} , and the duration of operation during which leakage occurs, T .

The dependence of both speed and energy dissipation on supply voltage creates a tension in circuit design: To make a system fast, the design must utilize high voltage levels, which increases energy demands; to make a system energy efficient, the design must utilize low voltage levels, which reduces circuit performance.

Dynamic voltage scaling has emerged as a powerful technique to reduce circuit energy demands. In a DVS system, the application or operating system identifies periods of low processor utilization that can tolerate reduced frequency. With reduced frequency, similar reductions are possible in the supply voltage. Since dynamic power scales quadratically with supply voltage, DVS technology can

Figure 1. Critical voltage and margins to meet worst-case reliability requirements.



significantly reduce energy consumption with little impact on perceived system performance.²

ERROR-TOLERANT DVS

Razor is an error-tolerant DVS technology. Its error-tolerance mechanisms eliminate the need for voltage margins that designing for “always-correct” circuit operations requires. The improbability of the worst-case conditions that drive traditional circuit design underlies the technology.

Voltage margins

Figure 1 shows margins for factors that can affect the voltage required to reliably operate a processor’s underlying circuitry for a given frequency setting. First, of course, the voltage must be sufficiently high to fully evaluate the longest circuit computation path in a single clock cycle. Circuit designers typically use static circuit-level timing analysis to identify this *critical voltage*.

To the critical voltage, they add the following voltage margins to ensure that all circuits operate correctly even in the worst-case operating environment:

- *Process margins* ensure that performance uncertainties resulting from manufacturing variations in transistor dimensions and composition do not prevent slower devices from completing evaluation within a clock cycle. Designers find the margin necessary to accommodate slow devices by using pessimistically slow devices to evaluate the critical path’s latency.
- *Ambient margins* accommodate slower circuit operations at high temperatures. The margin ensures correct operation at the worst-case temperature, which is typically 85-95° C.
- *Noise margins* safeguard against a variety of noise sources that introduce uncertainty in supply and signal voltage levels, such as di/dt noise in the supply voltage and cross-coupling noise in logic signals.

The sum of these voltages defines the minimum supply voltage that ensures correct circuit operation in even the most adverse conditions.

Worst-case improbability

In a simple experiment, we quantified the circuit error rates of an 18 × 18-bit multiplier block within a high-density field-programmable gate array. We used a Xilinx XC2V250-F456-5 FPGA because it contains full-custom multiplier blocks, which permit error-rate measurement with minimal routing-fabric overhead.

Test setup. Figure 2 shows the test harness and circuit schematic. The multiplier produces a 36-bit result each clock cycle. During FPGA logic placement, we directed synthesis to aggressively optimize the fast multiplier pipeline’s performance. The resulting placement was fairly efficient; the Xilinx static timing analyzer indicated that 82 percent of the fast multiplier stage latency was in the custom multiplier block.

Each cycle, two 48-bit linear feedback shift registers (LFSRs) generate 18-bit uncorrelated random values, sending them to a fast multiplier pipeline and, in alternating cycles, to two slow multiplier pipelines. The slow pipelines take turns safely computing the fast pipeline’s results, using a clock period that is twice as long as the fast multiplier pipeline.

As voltage decreases, values latched into the fast multiplier output latch may become *metastable*—that is, the values captured by the latch may be in transition between logic-0 and logic-1 and, thus, possess a voltage between these two well defined values. The empty stage after the fast multiplier stage (labeled “Stabilize” in Figure 2) gives these potentially metastable values time to stabilize back to 0 or 1 before they are compared with the known-correct slow multiplier results.

A multiplexer on the output of the slow pipelines selects the correct result to compare with the fast pipeline’s output. If the fast pipeline and slow pipeline results don’t match, a circuit timing error has occurred, and the error counter is incremented.

We used the Xilinx static timing analyzer to evaluate the design’s performance. The analyzer indicated that at 1.5 V and 85° C, the fast multiplier stage could run at up to 83.5 MHz; at 1.5 V and 27° C (room temperature), it could run at 88.6 MHz. All other support circuitry used to analyze multiplier errors was validated to 140 MHz. Thus, we are confident that all errors experienced in these experiments are localized to the fast multiplier pipeline circuits.

Error rates. Figure 3 illustrates the relationship between voltage and error rates for an 18 × 18-bit

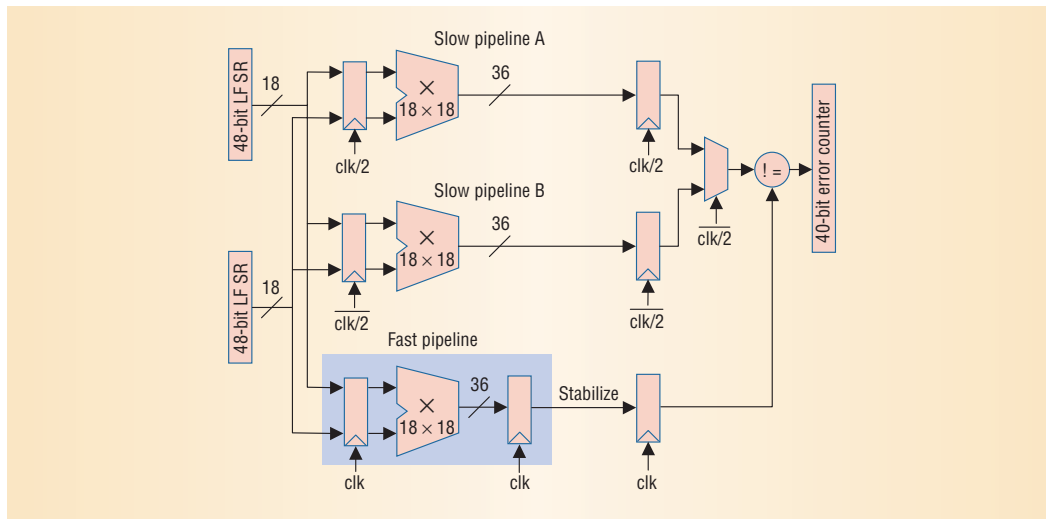


Figure 2. Error-rate test for 18 x 18-bit multiplier block. Shaded area in the test schematic indicates the multiplier circuit under test.

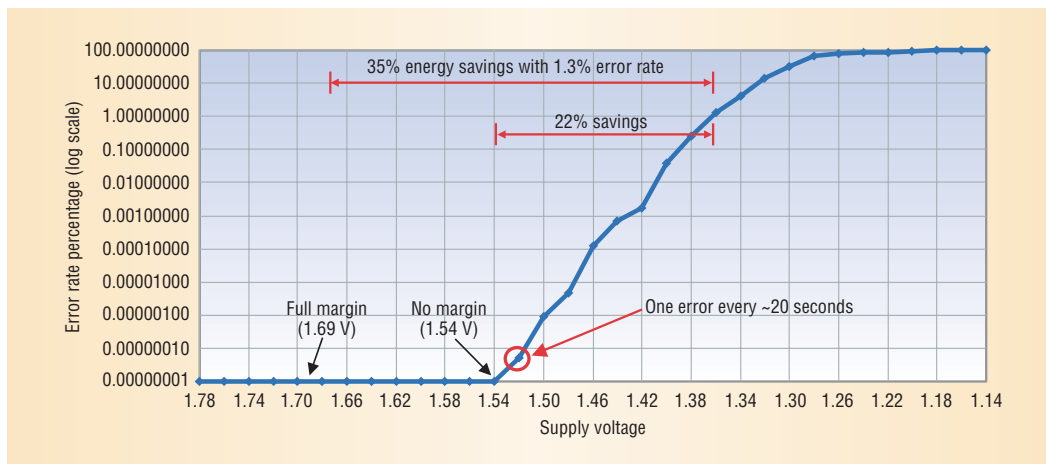


Figure 3. Measured error rates for an 18 x 18-bit FPGA multiplier block at 90 MHz and 27°C.

multiplier block running with random input vectors at 90 MHz and 27°C. The error rates are given as a percentage on a log scale.

The graph also shows two important design points:

- *no margin*—the lowest voltage that can still guarantee error-free circuit operation at 27°C, and
- *full margin*—the voltage at which the circuit runs without errors at 85°C in the presence of worst-case process variation and signal noise.

Traditional fault-avoidance design methodology sets the circuit voltage at the full margin point.

As Figure 3 shows, the multiplier circuit fails quite gracefully, taking nearly 180 mV to go from the point of the first error (1.54 V) to an error rate of 1.3 percent (1.36 V). At 1.52 V, the error rate is approximately one error every 20 seconds—or one error per 1.8 billion multiply operations.

The gradual rise in error rate is due to the dependence between circuit inputs and evaluation latency. Initially, only circuit inputs that require a complete

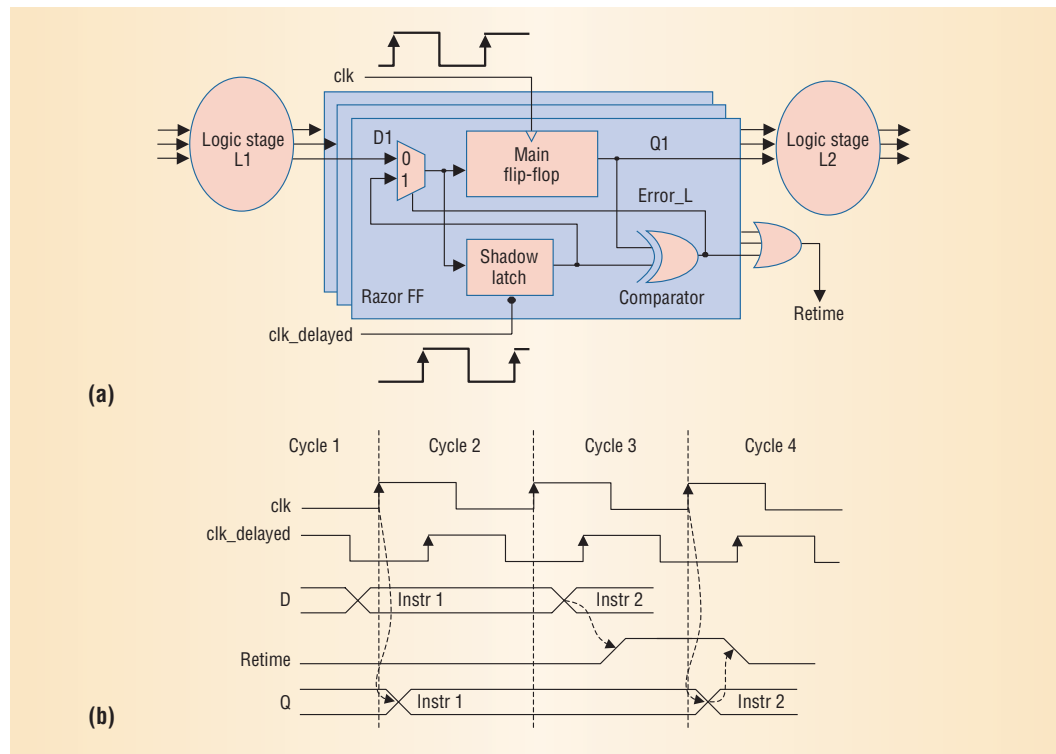
critical-path reevaluation result in a timing error. As the voltage continues to drop, the number of internal multiplier circuit paths that cannot complete within the clock cycle increases, along with the error rate. Eventually, voltage drops to the point where none of the circuit paths can complete in the clock period, and the error rate reaches 100 percent.

Clearly, the worst-case conditions are highly improbable. The circuit under test experienced no errors until voltage has dropped 150 mV (1.54 V) below the full margin voltage. If a processor pipeline can tolerate a small rate of multiplier errors, it can operate with a much lower supply voltage. For instance, at 330 mV below the full margin voltage (1.36 V), the multiplier would complete 98.7 percent of all operations without error, for a total energy savings (excluding error recovery) of 35 percent.

RAZORED PROCESSOR ARCHITECTURE

Given the improbability of worst-case operating conditions, an opportunity exists to reduce voltage commensurate with typical operating conditions. The processor pipeline must, however, incorporate

Figure 4. Razor flip-flop for a pipeline stage. (a) A shadow latch controlled by a delayed clock augments each flip flop. (b) Razor flip-flop operation with a timing error in cycle 2 and recovery in cycle 4.



a timing-error detection and recovery mechanism to handle the rare cases that require a higher voltage. In addition, the system must include a voltage control system capable of responding the operating condition changes, such as temperature, that might require higher or lower voltages.

Detecting circuit timing errors with Razor flip-flops

Figure 4a illustrates a Razor flip-flop for a pipeline stage. At the circuit level, a *shadow latch* augments each delay-critical flip-flop. A delayed clock controls the shadow latch.

Figure 4b illustrates a Razor flip-flop operation. In clock cycle 1, the combinational logic L1 meets the setup time by the clock's rising edge, and both the main flip-flop and the shadow latch will latch the same data. In this case, the error signal at the XOR gate's output remains low and the pipeline's operation is unaltered. In cycle 2, the combinational logic exceeds the intended delay due to subcritical voltage operation. In this case, the main flip-flop does not latch the data; but since the shadow latch operates using a delayed clock, it successfully latches the data in cycle 3.

To guarantee that the shadow latch will always latch the input data correctly, the allowable operating voltage is constrained at design time such that under worst-case conditions, the logic delay does not exceed the shadow latch's setup time. In cycle 3, a comparison of the valid shadow latch data with the main flip-flop data generates an error signal. In cycle 4, the shadow latch's data moves into the

main flip-flop and becomes available to the next pipeline stage L2.

If a timing error occurs in a particular clock cycle of pipeline stage L1, the data in L2 in the following clock cycle is incorrect and must be flushed from the pipeline. However, since the shadow latch contains stage L1's correct output data, the pipeline does not need to reexecute the instruction through L1. This is a key feature of Razor: It reexecutes an instruction failure in one pipeline stage through the *following* stage, while incurring a one-cycle penalty. The proposed approach therefore guarantees an instruction's forward progress and avoids the perpetual reexecution of an instruction at a particular pipeline stage because of timing failure.

Razor flip-flop construction must minimize the power and delay overhead. The power overhead is inherently low because in most cycles a flip-flop's input will not transition; thus, the only power overhead incurred comes from switching the delayed clock. To minimize even this power requirement, Razor inverts the main clock to generate the delayed clock locally, thus reducing its routing capacitance.

Many noncritical flip-flops in a design will not need Razor technology. For example, if the maximum delay at a flip-flop input is guaranteed to meet the required cycle time under the worst-case subcritical voltage setting, it isn't necessary to replace it with a Razor flip-flop because it will never need to initiate timing recovery. In the prototype Razor pipeline designed to study this problem, for example, we found that only 192 of a total of 2,408 flip-flops required Razor.

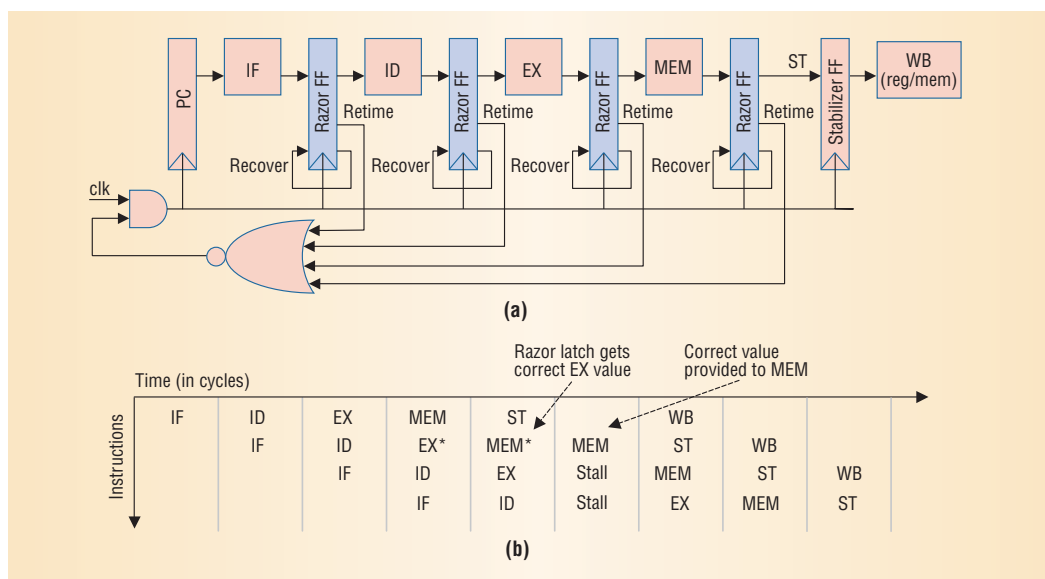


Figure 5. Pipeline recovery using global clock gating. (a) Pipeline organization and (b) pipeline timing for an error occurring in the execute (EX) stage. Asterisks denote a failing stage computation. IF = instruction fetch; ID = instruction decode; MEM = memory; WB = writeback.

Recovering pipeline state after timing-error detection

A pipeline recovery mechanism guarantees that any timing failures that do occur will not corrupt the register and memory state with an incorrect value. We have developed two approaches to recovering pipeline state.¹ The first is a simple method based on clock gating, while the second is a more scalable technique based on counterflow pipelining.³

Figure 5 illustrates pipeline recovery using a global clock-gating approach. In the event that any stage detects a timing error, pipeline control logic stalls the entire pipeline for one cycle by gating the next global clock edge. The additional clock period allows every stage to recompute its result using the Razor shadow latch as input. Consequently, recovery logic replaces any previously forwarded errant values with the correct value from the shadow latch.

Because all stages reevaluate their result with the Razor shadow latch input, a Razor flip-flop can tolerate any number of errant values in a single cycle and still guarantee forward progress. If all stages fail each cycle, the pipeline will continue to run but at half the normal speed.

In aggressively clocked designs, implementing global clock gating can significantly impact processor cycle time. Consequently, we have designed and implemented a fully pipelined recovery mechanism based on counterflow pipelining techniques. Figure 6 illustrates this approach, which places negligible timing constraints on the baseline pipeline design at the expense of extending pipeline recovery over a few cycles.

When a Razor flip-flop generates an error signal, pipeline recover logic must take two specific actions. First, it generates a *bubble* signal to nullify the computation in the following stage. This signal indicates to the next and subsequent stages that the

pipeline slot is empty. Second, recovery logic triggers the *flush train* by asserting the ID of the stage generating the error signal. In the following cycle, the Razor flip-flop injects the correct value from the shadow latch data back into the pipeline, allowing the errant instruction to continue with its correct inputs.

Additionally, the flush train begins propagating the failing stage's ID in the opposite direction of instructions. At each stage that the active flush train visits, a bubble replaces the pipeline stage. When the flush ID reaches the start of the pipeline, the flush control logic restarts the pipeline at the instruction *following* the failing instruction.

In the event that multiple stages generate error signals in the same cycle, all the stages will initiate recovery, but only the failing instruction closest to the end of the pipeline will complete. Later recovery sequences will flush earlier ones.

RAZOR PIPELINE PROTOTYPE

To obtain a realistic prediction of the power overhead for detecting and correcting circuit timing errors, we implemented Razor in a simplified 64-bit Alpha pipeline design, using Taiwan Semiconductor Manufacturing Co. 0.18-micrometer technology to produce the layout.¹ In addition to gate- and circuit-level power analysis on the error-detection-and-recovery design, we performed architectural simulations to analyze the overall throughput and power characteristics of Razor-based

voltage reduction for different benchmark test programs. The benchmark studies demonstrated that, on average, Razor reduced simulated power consumption by nearly a factor of two—a greater than 40 percent reduction—compared to traditional design-time dynamic voltage scaling and delay chain-based approaches.

Figure 6. Pipeline recovery using counterflow pipelining. (a) Pipeline organization and (b) pipeline timing for an error occurring in the execute (EX) stage. Asterisks denote a failing stage computation. IF = instruction fetch; ID = instruction decode; MEM = memory; WB = writeback.

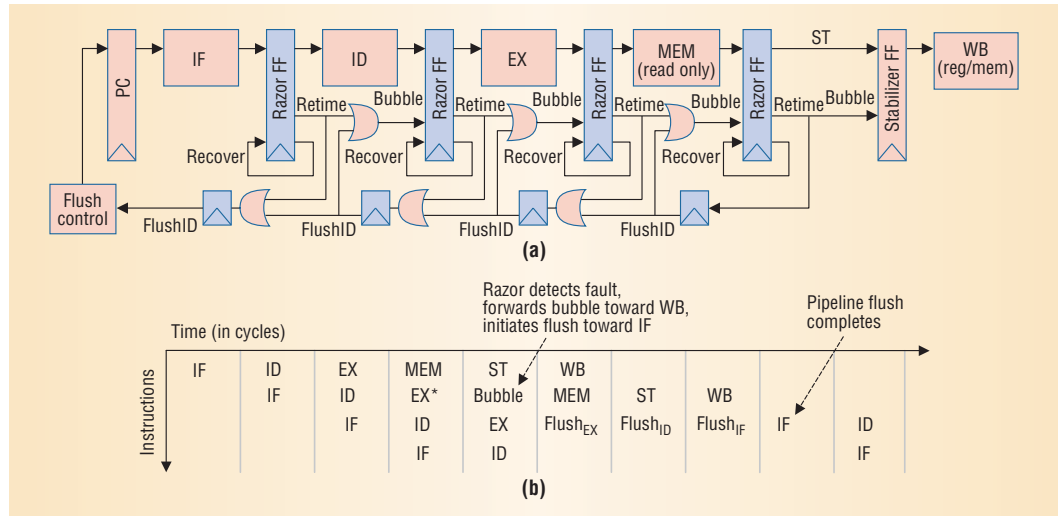
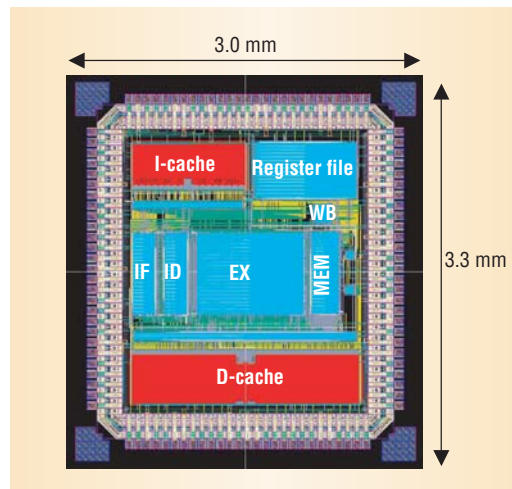


Figure 7. Razor prototype layout. A simplified 64-bit Alpha pipeline consists of instruction fetch, instruction decode, execute, and memory/writeback with both I- and D-cache.



Power analysis

Figure 7 shows the design layout; Table 1 lists the specifications and test results for error-free operation and for error-correction-and-recovery overhead. The pipeline consists of instruction fetch, instruction decode, execute, and memory/writeback with 8 Kbytes of both I-cache and D-cache. Performance analysis revealed that only the instruction decode and execute stages were critical at the worst-case voltage and frequency settings, thus requiring Razor flip-flops for their critical paths. While the overall design included a total of 2,408 flip-flops, only 192 of them implemented Razor technology. The clock for the Razor flip-flops was delayed by a half cycle from the system clock.

We performed both gate-level power simulations and SPICE (simulation program with integrated circuit emphasis) to evaluate the power overhead of the timing-failure detection and recovery circuit. The total power consumption during error-free operation at 200 MHz is 425 mW at 1.8 V.

Table 1 lists two energy consumption values, *switching* and *static*, for standard and Razor flip-

flops over one clock cycle in error-free operation. These values reflect whether the latched data is changing or not changing, respectively. We expect a power overhead of 12.2 mW for inserting delay buffers to meet short-path constraints, bringing the total overhead for the detection and recovery circuitry in error-free operation to 3.1 percent of total power consumption.

The energy required to detect a setup violation, generate an error signal, and restore the correct shadow latch data into the main flip-flop was 210 femtojoules (10^{-15}) per such event for each Razor flip-flop. The total energy required to perform a single timing error detection and recovery event in the pipeline was 189 picojoules (10^{-12}), resulting in an additional overhead of approximately 1 percent more total power when operating at a pessimistic 10 percent error rate.

Architectural benchmark tests

To further explore the design's efficiency, we developed an advanced simulation technique based on the SimpleScalar architectural tool set. This technique combines function-level architectural simulation with detailed SPICE-level circuit simulation, enabling the study of how voltage influences the timing of the pipeline stage computation.

Table 2 lists simulation results for the SPEC2000 benchmarks running on the simulated Razor prototype pipeline. Through extensive simulation, we identified the *fixed energy-optimal supply voltage* for each benchmark. This is the single voltage that results in the lowest overall energy requirement for each program. Table 2 also shows the average pipeline error rate, energy reduction, and pipeline throughput reduction (instructions per clock) at the fixed energy-optimal voltage. The total energy computation includes computation, Razor latch and check circuitry, and the total pipeline recovery energy incurred when an error is detected.

The Razor latches and error-detection circuitry

increase adder energy by about 4.3 percent. The energy for error detection and recovery is conservatively estimated at 18 times the cost of a single add (at 1.8 V), based on a six-cycle recovery sequence at typical activity rates.

Clearly, running the pipeline at a low error rate can reclaim significant energy. All of the benchmarks showed significant energy savings, ranging from 23.7 to 64.2 percent. One particularly encouraging result is that Razor mutes error rates and performance impacts up to and slightly past the energy-optimal voltage, after which the error rate rises very quickly. At the energy-optimal voltage point, the benchmarks suffered at most a 2.49 percent reduction in pipeline performance due to recovery flushes.

There appears to be little trade-off in performance when fully exploiting energy savings at sub-critical voltages. We have simulated voltages down to 0.6 V, but our Razor prototype design can only validate circuit timing down to 1.2 V. This constraint will limit the energy savings of four of the benchmarks.

Since additional voltage scaling headroom exists, we are examining techniques to further reduce voltage in future prototype designs.

FUTURE WORK

We submitted the prototype Razor pipeline design for fabrication in October 2003 and expect to test the real silicon early this year.

Meanwhile, two immediate questions must be answered to fully implement Razor technology: How do we design razored control logic, and how do we design razored memories? Our initial

Table 1. Razor prototype specifications.

Description	Specification
Technology node	0.18mm
Voltage range	1.8 V to 1.2 V
Total number of logic gates	45,661
D-cache size	8 Kbytes
I-cache size	8 Kbytes
Die size	3 × 3 mm
Clock frequency	200 MHz
Clock delay	2.5 nS
Total number of flip-flops	2,408
Number of Razor flip-flops	192
Total number of delay buffers	2,498
<i>Error-free operation</i>	
Total power	425 mW
Standard FF energy (switching/static)	49 fJ / 95 fJ
Razor FF energy (switching/static)	60 fJ / 160 fJ
Total delay buffer power overhead	12.2 mW
Total power overhead	3.1%
<i>Error correction and recovery overhead</i>	
Energy per Razor FF per error event	210 fJ
Total energy per error event	189 pJ
Recovery power overhead at 10% error rate	1%

research indicates that we can develop microarchitectural solutions to address delay failures that occur in the control logic, and we are investigating the use of double-sampling sense amplifiers for developing a Razor-enabled cache.

We see several potential applications of Razor technology in the future.

Self-tuning systems

In its current form, Razor sets voltage globally—chip wide, but we could refine it to allow distributed voltage control. Under a distributed control system, each processor pipeline stage could operate at a separate, potentially different voltage deter-

Table 2. Energy-optimal characteristics for SPEC2000 benchmarks.

Program	Optimal V_{dd}	Error rate (percent)	Energy reduction (percent)	Pipeline throughput reduction (percent)
bzip	1.1	0.31	57.6	0.70
crafty	1.175	0.41	60.5	0.60
con	1.3	1.21	34.4	1.24
gap	1.275	1.15	30.1	2.49
gcc	1.375	1.62	23.7	1.47
gzip	1.3	1.03	35.6	0.41
mcf	1.175	0.67	48.7	0.00
parser	1.2	0.61	47.9	0.29
twolf	1.275	2.67	30.7	0.31
vortex	1.3	0.53	42.8	0.14
vpr	1.075	0.01	64.2	0.00
Average			42.4	

Meeting power challenges, especially for mobile systems will require sustained rule-breaking innovation.

mined by monitoring pipeline stage error rates. Releasing the constraint of a single operating voltage enables significant optimizations of voltage assignments across the processor stages, leading to further power savings.

Alternatively, we can maintain global voltage control but skew the clock phase individually for each unit to perform a type of dynamic retiming. High-clock-rate processors employ similar techniques to statically adjust clock skew.

Extreme voltage scaling

Current voltage-scalable designs are typically limited to operating voltages within 50 percent of maximum supply voltage.^{4,5} This translates to a total power improvement of at most four times, due to the quadratic dependence of power on voltage.

However, our work on drowsy caches shows that memories can operate as low as the threshold voltage of their transistors (typically one-third of normal supply voltage).^{6,7} In fact, it is possible to push the supply voltage to subthreshold levels as low as a few hundred millivolts. The power savings possible in such regimes is dramatic—approaching a factor of 10. The cost of bringing units out of drowsy mode when they are needed is an obstacle to this approach, but we have already solved many of the issues in this area. Operating at these levels introduces a degree of uncertainty in unit behaviors, which makes subthreshold voltage scaling an ideal application for Razor.

Reliability

Razor technology and extensions to it may help solve more general transient failures. For example, a number of radiation sources in nature can affect electronic circuit operations. The two most prevalent are

- *gamma rays*, which arrive from space; while the atmosphere filters out most of them, some occasionally reach the Earth's surface, especially at higher altitudes; and
- *alpha particles*, which are created when atomic impurities (found in all materials) decay.

When these energetic particles strike a very small transistor, they can deposit or remove sufficient charge to temporarily turn the device on or off, possibly creating a logic error.^{8,9} They have posed a problem for dynamic RAM designs since the late 1970s when DRAM capacitors became sufficiently

small to be affected by them.¹⁰

It is difficult to shield against natural radiation sources. Gamma rays that reach the Earth's surface have such high momentum that only thick, dense materials can stop them.¹¹ A thin shield can stop alpha particles, but if the shield is not free of atomic impurities, it becomes an additional source of natural radiation. Neither shielding approach is cost effective for most system designs.

Furthermore, the smaller feature sizes that have driven the digital revolution make the particles relatively larger. Their impact (literally) is growing dramatically, and designers will likely be forced to adopt fault-tolerant design solutions to protect against them. Razor offers a solution that may compare well with conventional error-correcting codes.

Process variability

As feature sizes drop below 100 nanometers, the variation in key parameters, such as supply and threshold voltages and transistor widths, increases dramatically. These variations limit the guaranteed performance of circuits and potentially to neutralize the benefits of smaller silicon geometries.

Razor removes the supply voltage design margins normally needed to account for worst-case technology variations between different chip instances (fabrication-time variability). Razor designs can also adjust dynamically to a computation's data-dependent nature, saving energy by lowering voltage when data dependencies permit (runtime variability).

Power is the next great challenge for computer systems designers, especially those building mobile systems with frugal energy budgets. We believe that meeting this challenge will require sustained rule-breaking innovation. Technologies like Razor enable “better than worst-case design,” opening the door to methodologies that optimize for the common case rather than the worst.

Optimizing designs to meet the performance constraints of worst-case operating points requires enormous circuit effort, resulting in tremendous increases in logic complexity and device sizes. It is also power-inefficient because it expends tremendous resources on operating scenarios that seldom occur. Using recomputation to process rare, worst-case scenarios leaves designers free to optimize standard cells or functional units—at both their architectural and circuit levels—for the common case, saving both area and power. ■

Acknowledgments

This work was supported by ARM, an Intel Graduate Fellowship, the Defense Advanced Research Projects Agency, the Semiconductor Research Corporation, the Gigascale Silicon Research Center, and the National Science Foundation.

References

1. D. Ernst et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," *Proc. 36th Ann. Int'l Symp. Microarchitecture (MICRO-36)*, IEEE CS Press, 2003; pp. 7-18.
2. T. Mudge, "Power: A First-Class Architectural Design Constraint," *Computer*, Apr. 2001, pp. 52-58.
3. R.F. Sproull, I.E. Sutherland, and C.E. Molnar, "The Counterflow Pipeline Processor Architecture," *IEEE Design and Test of Computers*, Fall 1994, pp. 48-59.
4. K. Flautner and T. Mudge, "Vertigo: Automatic Performance-Setting for Linux," *Proc. 5th Conf. Operating Systems Design and Implementation (OSDI)*, ACM Press, 2002, pp. 105-116.
5. K. Flautner, S. Reinhardt, and T. Mudge, "Automatic Performance Setting for Dynamic Voltage Scaling," *ACM J. Wireless Networks*, Sept. 2002, pp. 507-520.
6. N. Kim et al., "Drowsy Instruction Caches: Leakage Power Reduction Using Dynamic Voltage Scaling and Cache Subbank Prediction," *Proc. 35th Ann. IEEE/ACM Symp. Microarchitecture (MICRO-35)*, IEEE CS Press, 2002, pp. 219-230.
7. K. Flautner et al., "Drowsy Caches: Simple Techniques for Reducing Leakage Power," *Proc. 29th Ann. Int'l Symp. Computer Architecture*, IEEE Press, 2002, pp. 148-157.
8. J.Z. et al., "IBM Experiments in Soft Fails in Computer Electronics," *IBM J. Research and Development*, Jan. 1996, pp. 3-18.
9. P. Rubinfeld, "Managing Problems at High Speed," *Computer*, Jan. 1998, pp. 47-48.
10. T. May and M. Woods, "Alpha-Particle-Induced Soft Errors in Dynamic Memories," *IEEE Trans. Electron Devices*, vol. 26, no. 2, 1979, pp. xx-yy.
11. J. Ziegler, "Terrestrial Cosmic Rays," *IBM J. Research and Development*, Jan. 1996, pp. 19-39.

Todd Austin is an associate professor of electrical engineering and computer science at the University of Michigan. His research interests include computer architecture, compilers, computer system verification, and performance analysis tools and techniques. Austin received a PhD in computer science from the University of Wisconsin. Contact him at austin@umich.edu.

David Blaauw is an associate professor of electrical engineering and computer science at the University of Michigan. His research interests include VLSI design and CAD with emphasis on circuit analysis and optimization problems for high-performance and low-power microprocessor designs. Blaauw received a PhD in computer science from the University of Illinois, Urbana-Champaign. Contact him at blaauw@umich.edu or visit his Web site at <http://blaauw.eecs.umich.edu>.

Trevor Mudge is the Bredt Professor of Electrical Engineering and Computer Science at the University of Michigan. In addition, he runs Idiot Savants, a chip-design consultancy. His research interests include computer architecture, CAD, and compilers. Mudge received a PhD in computer science from the University of Illinois, Urbana-Champaign. He is a Fellow of the IEEE and a member of the ACM, the IEE, and the British Computer Society. Contact him at tmm@eecs.umich.edu.

Krisztián Flautner is the director of advanced research at ARM Ltd. and the architect of ARM's Intelligent Energy Management technology. His research interests include high-performance, low-power processing platforms to support advanced software environments. Flautner received a PhD in computer science and engineering from the University of Michigan. Contact him at krisztian.flautner@arm.com.