Appears in the 36th Annual International Symposium on Microarchitecture (MICRO-36), December 2003.

Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation

Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham,
Conrad Ziesler, David Blaauw, Todd Austin, Krisztian Flautner[1], and Trevor Mudge

Advanced Computer Architecture Lab
The University of Michigan
1301 Beal Ave
Ann Arbor, MI 48109
razor@eecs.umich.edu

ARM Ltd [1]
110 Fulbourn Road
Cambridge, UK CB1 9NJ
krisztian.flautner@arm.com

Abstract

With increasing clock frequencies and silicon integration, power aware computing has become a critical concern in the design of embedded processors and systems-on-chip. One of the more effective and widely used methods for power-aware computing is dynamic voltage scaling (DVS). In order to obtain the maximum power savings from DVS, it is essential to scale the supply voltage as low as possible while ensuring correct operation of the processor. The critical voltage is chosen such that under a worst-case scenario of process and environmental variations, the processor always operates correctly. However, this approach leads to a very conservative supply voltage since such a worst-case combination of different variabilities will be very rare. In this paper, we propose a new approach to DVS, called Razor, based on dynamic detection and correction of circuit timing errors. The key idea of Razor is to tune the supply voltage by monitoring the error rate during circuit operation, thereby eliminating the need for voltage margins and exploiting the data dependence of circuit delay. A Razor flip-flop is introduced that double-samples pipeline stage values, once with a fast clock and again with a time-borrowing delayed clock. A metastability-tolerant comparator then validates latch values sampled with the fast clock. In the event of a timing error, a modified pipeline mispeculation recovery mechanism restores correct program state. A prototype Razor pipeline was designed in 0.18 μm technology and was analyzed. Razor energy overheads during normal operation are limited to 3.1%. Analyses of a full-custom multiplier and a SPICE-level Kogge-Stone adder model reveal that substantial energy savings are possible for these devices (up to 64.2%) with little impact on performance due to error recovery (less than 3%).

1 Introduction

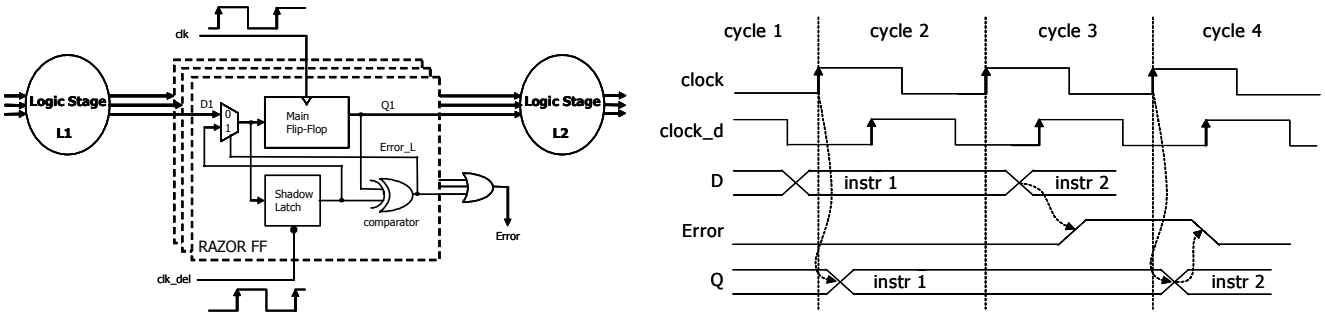
A critical concern for embedded systems is the need to deliver high levels of performance given ever-diminishing power budgets. This is evident in the evolution of the mobile phone: in the last 7 years mobile phones have shown a 50X improvement in talk-time per gram of battery[1], while at the same time taking on new computational tasks that only recently appeared on desktop computers, such as 3D graphics, audio/video, internet access, and gaming. As the breadth of applications for these devices widens, a single operating point is no longer sufficient to efficiently meet their processing and power consumption requirements. For example, MPEG video playback requires an order-of-magnitude higher performance than playing MP3s. However, running at the performance level necessary for video is energy-inefficient for audio. The gap between high performance and low power can be bridged through the use of dynamic voltage scaling (DVS) [16], where periods of low processor utilization are exploited by lowering the clock frequency to the minimum required level, allowing corresponding reduction in the supply voltage. Since dynamic energy scales quadratically with supply voltage, significant reduction in energy use can be obtained [14].

Enabling systems to run at multiple frequency and voltage levels is a challenging process and requires characterization of the processor to ensure that its operation remains correct at the required operating points. The minimum possible supply voltage that results in correct operation is referred to as the *critical* supply voltage. The critical supply voltage must be sufficient to ensure correct operation in the face of a number of environmental and process related variabilities that can impact circuit performance. These include unexpected voltage drops in the power supply network, temperature fluctuations, gate-length and doping concentration variations, cross-coupling noise, etc. These variabilities may be data dependent, meaning that they exhibit their worst-case impact on circuit performance only under certain instruction and data sequences, and are composed of both local and global components. For instance, local process variations will impact specific regions of the die in different and independent ways, while global process variation impacts the circuit performance of the entire die and creates variation from one die to the next. Similarly, temperature and supply drop have local and global components, while cross-coupling noise is a predominantly local effect.

To ensure correct operation under all possible variations, a conservative supply voltage is typically selected at design-time using corner analysis. Hence, margins are added to the critical voltage to account for uncertainty in the circuit models and to account for the worst-case combination of variabilities. However, such a worst-case combination of variabilities may be very rare or even impossible in a particular instance of a chip making this approach overly conservative. And, with process scaling, the environmental and process variabilities are expected to increase, worsening the required voltage margins.

To allow for more aggressive power reduction, the supply voltage can be tuned to an individual processor chip using embedded inverter delay chains [5]. The delay of the inverter chain is used as a prediction of the critical path delay of the circuit and the supply voltage is tuned during processor operation to meet a predetermined delay through the inverter-chain. This approach to DVS has the advantage that it dynamically adjusts the operating voltage to account for global variations in supply voltage drop, temperature fluctuation, and process variations. However, it cannot account for local variations, such as local supply voltage drops, intra-die process variations, and cross-coupled noise, and therefore requires the addi-

1. Comparison of standard configurations of Nokia 232 and Ericsson T68 phones.



(a) **(b)**

Figure 1. Pipeline augmented with Razor latches and control lines.

tion of safety margins to the critical voltage. Also, the delay of an inverter chain does not scale with voltage and temperature in the same way as the delays of the critical paths of the actual design, which can contain complex gates and pass-transistor logic, which again necessitate extra voltage safety margins. In future technologies, the local component of environmental and process variation is expected to become more prominent and, as noted in [6], the sensitivity of circuit performance to these variations is higher at lower operating voltages, thereby increasing the necessary margins and reducing the scope for energy savings.

In this paper, we propose a new approach to DVS, referred to as *Razor*, which is based on dynamic detection and correction of speed path failures in digital designs. The key idea of Razor is to tune the supply voltage by monitoring the error rate during operation. Since this error detection provides *in-situ* monitoring of the actual circuit delay, it accounts for both global and local delay variations and does not suffer from voltage scaling disparities. It therefore eliminates the need for voltage margins that are necessary for "always-correct" circuit operation in traditional designs. In addition, a key feature of Razor is that operation at sub-critical supply voltages does not constitute a catastrophic failure, but instead represents a trade-off between the power penalty incurred from error correction against additional power savings obtained from operating at a lower supply voltage.

It was previously observed that circuit delay is strongly data dependent, and only exhibits its worst-case delay for very specific instruction and data sequences [24]. From this it can be conjectured that for moderately sub-critical supply voltages only a few critical instructions will fail, while a majority of instructions will continue to operate correctly. Our hardware measurements and circuit simulation studies support this conjecture and demonstrate that the circuit operation degrades gracefully for sub-critical supply voltages, showing a gradual increase in the error rate. The proposed Razor approach automatically exploits this data-dependence of circuit delay by tuning the supply voltage to obtain a small, but non-zero error rate. It was found that if the error rate is maintained sufficiently low, the power overhead from error correction is minimal, while substantial power savings are obtained due to operating the circuit at a lower supply voltage. Note that as the processor executes different sets of instructions, the supply voltage automatically adjusts to the delay characteristics of the executed instruction sequence, lowering the supply voltage for instruction sequences with many non-critical instructions, and raising the supply voltage for instruction sequences that are more delay intensive.

We propose a combination of circuit and architectural techniques for low cost in-situ error detection and correction of delay failures. At the circuit level, each delay-critical flip-flop is augmented with a so-called *shadow latch* which is controlled using a delayed clock. The operating voltage is constrained such that the worst-case delay is guaranteed to meet the shadow latch setup time, even though the main flip-flop could fail. By comparing the values latched by the flip-flop and the shadow latch, a delay error in the main flip-flop is detected. The value in the shadow latch, which is guaranteed to be correct, is then utilized to correct the delay failure. We present several architectural solutions for error correction, ranging from simple clock gating to more sophisticated mechanisms that augment the existing mispeculation recovery infrastructure.

The proposed Razor technique was implemented in a prototype 64-bit Alpha processor design. This prototype implementation was used to obtain a realistic prediction of the power overhead for *in-situ* error correction and detection. We also studied the error-rate trends for datapath components using both circuit-level simulation as well as silicon measurements of a full-custom multiplier block. Architectural simulations were then performed to analyze the overall throughput and power characteristics of Razor based DVS for different benchmark test programs. We demonstrate that on average, Razor reduced simulated power consumption by more than 40%, compared to traditional design-time DVS and delay-chain based approaches.

The remainder of this paper is organized as follows. In Section 2, we present the implementation of Razor, providing a detailed description of both the proposed circuit and architectural techniques. In Section 3, we discuss the simulation framework for Razor-based DVS and present error rate studies and our simulation results. In Section 4 we present a detailed survey of prior work in DVS. Finally, in Section 5, we draw our conclusions.

2 Razor Error Detection/Correction

Razor relies on a combination of architectural and circuit level techniques for efficient error detection and correction of delay path failures. The concept of Razor is illustrated in Figure 1(a) for a pipeline stage. Each flip-flop in the design is augmented with a so-called *shadow latch* which is controlled by a delayed clock. We illustrate the operation of a Razor flip-flop in Figure 1(b). In clock cycle 1, the combinational logic $L1$ meets the setup time by the rising edge of the clock and both the main flip-flop and the shadow latch will latch the correct data. In this case, the error signal at the output of the XOR gate remains low and the operation of the pipeline is unaltered.

In cycle 2 in Figure 1(b), we show an example of the operation when the combinational logic exceeds the intended delay due to sub-critical voltage scaling. In this case, the data is not latched by the main flip-flop, but since the shadow-latch operates using a delayed clock, it successfully latches the data some time 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 setup time of the shadow latch. By comparing the valid data of the shadow latch with the data in the main flip-flop, an error signal is then generated in cycle 3 and in the subsequent cycle, cycle 4, the valid data in the shadow latch is restored into the main flip-flop and becomes available to the next pipeline stage $L2$. Note that the local error signals *Error_l* are OR'ed together to ensure that the data in all

flip-flops is restored even when only one of the Razor flip-flops generates an error.

If an error occurs in pipeline stage *L*1 in a particular clock cycle, the data in *L*2 in the following clock cycle is incorrect and must be flushed from the pipeline using one of the pipeline control methods described in Section 2.2. However, since the shadow latch contains the correct output data of pipeline stage *L*1, the instruction does not need to be re-executed through this failing stage. Thus, a key feature of Razor is that if an instruction fails in a particular pipeline stage it is re-executed through the *following* pipeline stage, while incurring a one cycle penalty. The proposed approach therefore *guarantees forward progress* of a failing instruction, which is essential to avoid the perpetual failure of an instruction at a particular stage in the pipeline.

In addition to invalidating the data in the following pipeline stage, an error must also stall the preceding pipeline stages while the shadow latch data is restored into the main flip-flops. A number of different methods, such as clock gating or flushing the instruction in the preceding stages, were examined to accomplish this and are discussed in Section 2.2. The proposed approach also raises a number of circuit related issues. The Razor flip-flop must be constructed such that the power and delay overhead is minimized. Also, the presence of the delayed clock introduces a new short-path constraint in the design. And finally, allowing the setup time of the main flip-flop to be exceeded raises the possibility of meta-stability. These issues are discussed in more detail in Section 2.1. In the proposed Razor based DVS approach, the error signal is used to tune the supply voltage to its optimal value. In Section 2.3, we therefore discuss different algorithms to control the supply voltage based on the observed error rate.

In general, maximum power savings is obtained from Razor technology when it is applied to all parts of a microprocessor design. To accomplish this, we identify three distinct design challenges. The first design challenge, and the focus of this paper, is the detection and recovery of timing errors in combinational logic contained within pipeline datapaths, e.g., adders, shifters, and decode logic. The second design challenge is the application of Razor to on-chip SRAM structures. In SRAM structures, such as register files and caches, it is necessary to introduce Razor-compatible sense amplifiers and support for fast non-speculative stores. The third challenge is the use of Razor on pipeline control logic to restore correct program execution in the presence of incorrect control decisions.

For the sake of brevity and clarity, the focus of this paper is limited to the first design challenge, which is the use of Razor on combinational logic blocks contained within the pipeline datapaths. We therefore apply Razor to a simple embedded processor which utilizes an in-order pipeline with simple control and small caches. In such a processor, control logic and SRAM structures remain error-free, even at the worst-case frequency and voltage and do not require Razor technology. However, to effectively apply Razor in large microprocessor designs with large caches and complex control logic, it will be necessary to apply Razor technology to all parts of the design. Therefore, in concert with the effort presented in this paper, we are developing Razor-compatible memory structures based on bit-line sampling and architectural modifications for reduced typical-case latency. For control logic, we are developing techniques to checkpoint control state to enable control logic recovery. These additional developments will be presented in future reports.

2.1 Circuit-level implementation issues

A key requirement for Razor based DVS is that during error-free operation, the delay and power overhead due to the error detection and correction circuitry is minimal. Otherwise, the power gain from more aggressive voltage scaling is overcome by the power overhead due to the presence of the error detection and correction circuitry. In addition, the overhead of performing an error correction must also be minimized to enable efficient operation at moderate

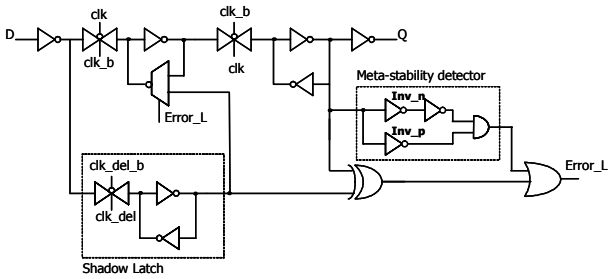


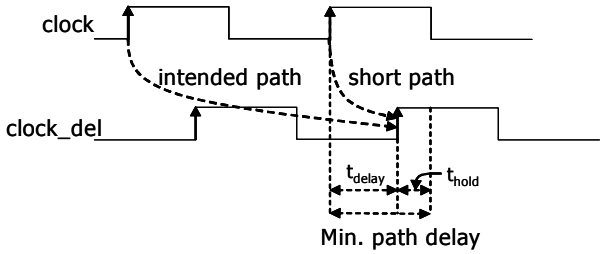
Figure 2. Reduced overhead Razor flip-flop and meta-stability detection circuits.

error rates. A number of methods were applied to reduce the power and delay overhead of the Razor flip-flop, shown in Figure 1. The multiplexer at the input the razor flip-flop results in a significant delay and power overhead, and was therefore moved to the feedback path of the master latch of the main flip-flop, as shown in Figure 2. Hence, it introduces only a slight increase in the capacitive loading of the critical path and has minimal impact on the performance and power of the design.

The power overhead of Razor is also reduced by the fact that in most cycles, the input of a flip-flop will not transition and only the power overhead from switching the delayed clock is incurred. To further minimize this additional clock power, the delayed clock is locally generated, reducing its routing capacitance. If the delayed clock is delayed by half the clock cycle, it can be derived by simply inverting the main clock. Also, many non-critical flip-flops in the design do not need Razor. If the maximum delay at the input of a flip-flop is guaranteed to meet the required cycle time under the worst-case sub-critical voltage, the flip-flop cannot fail and does not need to be replaced with a Razor flip-flop. It was found that in the prototype Alpha processor only 192 flip-flops out of a total of 2408 required Razor, thereby significantly reducing the power overhead of the Razor approach. For this prototype processor, the total power overhead in error free operation (due to Razor flip-flops) was found to be less than 1%, while the delay overhead was negligible.

The use of a delayed clock at the shadow latch raises the possibility that a short path in the combinational logic will corrupt the data in the shadow latch. Figure 3 shows how a short-path allows data launched at the start of a cycle to be latched into the shadow latch, instead of the data launched from the previous cycle. To prevent this corruption of the shadow latch data, a minimum-path length constraint is added at the input of each Razor flip-flop in the design. These minimum-path constraints result in the addition of buffers during logic synthesis to slow down fast paths and therefore introduce a certain power overhead. Figure 3 shows that the minimum-path constraint is equal to the clock delay t_{delay} plus the hold time t_{hold} of the shadow latch (which is typically a small negative value). A large clock delay increases the severity of the short path constraint and therefore increases the power overhead due to the need for additional buffers. On the other hand, a small clock delay reduces the margin between the main flip-flop and the shadow latch, and hence reduces the amount by which the supply voltage can be dropped below the critical supply voltage. The clock delay therefore presents a trade-off between the power overhead incurred from short-path correction and the degree of possible power saving from sub-critical voltage operation. In the prototype 64-bit Alpha design, the clock delay was set at 1/2 the clock period. This simplified the generation of the delayed clock while the short-path constraints could still be easily met and resulted in a power overhead (due to buffers) of less than 3%.

In subcritical voltage operation, it is possible that the data at the input of the main latch transitions at the same time as the clock. This can give rise to *meta-stability* of the main flip-flop, where the output



Min. Path Delay > t_{delay} + t_{hold}

Figure 3. Short Paths Constraints.

voltage does not resolve to a definite high or low voltage, but instead hovers near Vdd/2 [4]. The danger of meta-stability is that different fan-out gates may interpret this indeterminate voltage level as different logic states, or may even enter a meta-stable state themselves. It is important to note that, since the minimum sub-critical voltage is constrained such that the setup time of the shadow latch is always met, the shadow latch is stable and can not exhibit meta-stability. However, if the main flip-flop is meta-stable, it is impossible to determine if its latched value is correct or not using the XOR gate in Figure 2. Hence, we include a meta-stability detector circuit in the Razor flip-flop which detects the presence of a meta-stable voltage levels, as shown in Figure 2. A detected meta-stability event is corrected the same way as a regular delay failure, and results in the stable and correct data value from the shadow latch being restored in the main flip-flop. For simplicity, the meta-stability detector in Figure 2 is constructed using two inverter gates with different skewed P/N ratios, such that they switch at different voltage levels. If the two inverters interpret the result differently, the flip-flop voltage is not definitive and may be meta-stable. Note that, any suitable comparator circuit could be utilized and that these meta-stability events do not result in a failure of the system but are corrected using the existing Razor error correction infrastructure.

However, it is well known that complete system failure due meta-stability to cannot be completely avoided and only its probability of occurrence can be reduced to negligible levels [4]. In the proposed Razor design, this manifests itself in the small but finite probability that the error signal itself becomes meta-stable. This could occur if the main flip-flop output voltage was near the edge of the meta-stable voltage range and, hence, the meta-stability detector was unable to determine if a meta-stability event occurred or not. In this case, the error signal will not resolve to a definite voltage level and ambiguity will exist in the logic value of the error signal, possibly causing a failure in the error correction mechanism. A standard approach to reduce the probability of such an event to negligible levels is to double latch the signal. However, this would delay the detection of an error in the main flip-flop by one cycle, complicating the error recovery mechanism. We therefore employ at the same time an additional mechanism to detect metastable error signals, where the error signal is double latched using two skewed flip-flops. The probability that the outputs of the second set of flip-flops are meta-stable is hence reduced to a negligible level and by comparing their output values, the presence of a meta-stable error signal one cycle earlier can be reliably detected. Under normal operation, the error signal will resolve to a definite voltage level and the output values of the two skewed flip-flops will match, indicating that the performed error correction was executed correctly. However, in the unlikely event that the error signal is meta-stable, the outputs of the skewed latches will differ in the subsequent clock cycle indicating that the error correction was unsafe and could have failed. In this case, a so called *panic* signal is generated, which requires that the entire pipeline is flushed and restarted. In this case, guaranteed forward progress is lost, and the supply voltage level must be raised to avoid

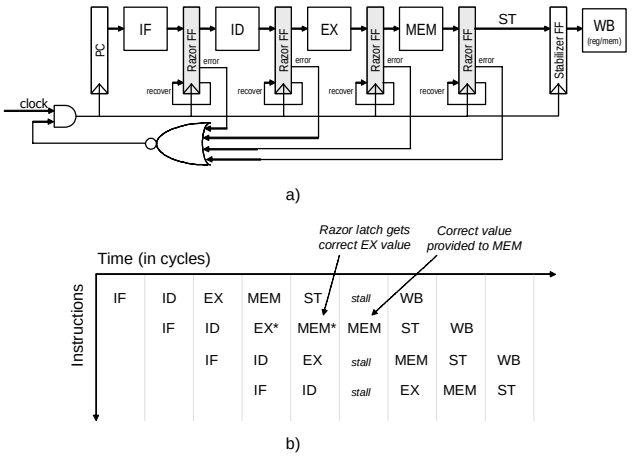


Figure 4. Pipeline recovery using global clock gating. Figure a) shows the pipeline organization, Figure b) illustrates the pipeline timing for a failure in the EX stage of the pipeline. The "*" denotes a failed stage computation.

possible perpetual failure of the same instruction. However, the possibility of a meta-stable error signal is extremely small and does not constitute a significant burden on the power and performance of the processor. Also, only one set of double latches is needed for each pipeline stage, meaning that the power overhead during error-free operation is negligible.

2.2 Pipeline error recovery mechanisms

The pipeline error recovery mechanism must guarantee that, in the presence of Razor errors, register and memory state is not corrupted with an incorrect value. In this section, we highlight two possible approaches to implementing pipeline error recovery. The first is a simple but slow method based on clock gating, while the second method is a much more scalable technique based on counterflow pipelining.

Recovery using clock gating. Figure 4(a) illustrates a simple approach to pipeline error recovery based on global clock gating. In the event that any stage detects a Razor error, the entire pipeline is stalled 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, any previously forwarded errant values will be replaced with the correct value from the Razor shadow latch. Since all stages re-evaluate their result with the Razor shadow latch input, any number of errors can be tolerated in a single cycle and forward progress is guaranteed. If all stages produce an error each cycle, the pipeline will continue to run, but at 1/2 the normal speed.

It is imperative that errant pipeline results not be written to architected state before it has been validated by Razor. Since validation of Razor values takes two additional cycles (i.e., one for error detection and one for panic detection), there must be two non-speculative stages between the last Razor latch and the writeback (WB) stage. In our design, memory accesses to the data cache are non-speculative, hence, only one additional stage labeled ST for stabilize is required before writeback (WB). The ST stage introduces an additional level of register bypass. Since store instructions must execute non-speculatively, they are performed in the WB stage of the pipeline.

Figure 4(b) gives a pipeline timing diagram of a pipeline recovery for an instruction that fails in the EX stage of the pipeline. The first failed stage computation occurs in the 4th cycle, when the second instruction computes an incorrect result in the EX stage of the pipeline. This error is detected in the 5th cycle, but only after the MEM stage has computed an incorrect result using the errant value

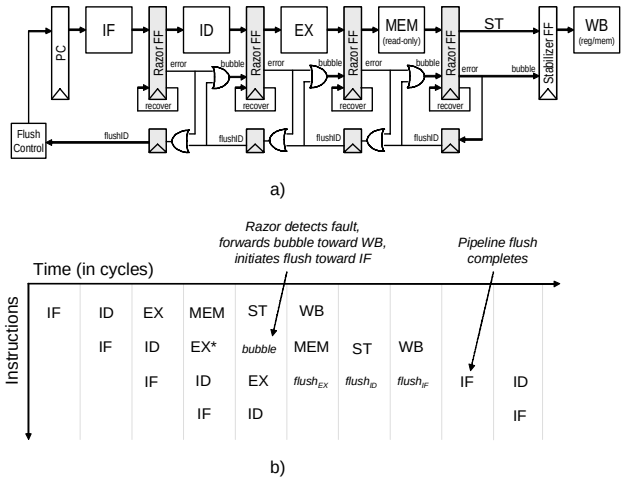


Figure 5. Pipeline recovery using counterflow pipelining. Figure a) shows the pipeline organization, Figure b) illustrates the pipeline timing for a failure in the EX stage of the pipeline. The "*" denotes a failed stage computation.

forward from the EX stage. After the error is detected, a global clock stall occurs in the 6th cycle, permitting the correct EX result in the Razor shadow latch to be evaluated by the MEM stage. In the 7th cycle, normal pipeline operation resumes.

Recovery using counterflow pipelining. In aggressively clocked designs, it may not be possible to implement global clock gating without significantly impacting processor cycle time. Consequently, we have designed and implemented a fully pipelined error recovery mechanism based on counterflow pipelining techniques [19]. The approach, illustrated in Figure 5(a), places negligible timing constraints on the baseline pipeline design at the expense of extending pipeline recovery over a few cycles. When a Razor error is detected, two specific actions must be taken. First, the errant stage computation following the failing Razor latch must be nullified. This action is accomplished using the *bubble* signal, which indicates to the next and subsequent stages that the pipeline slot is empty. Second, the *flush train* is triggered by asserting the stage ID of failing stage. In the following cycle, the correct value from the Razor shadow latch data is injected back into the pipeline, allowing the errant instruction to continue with its correct inputs. Additionally, the flush train begins propagating the ID of the failing stage in the opposite direction of instructions. At each stage visited by the active flush train, the corresponding pipeline stage and the one immediately preceding are replaced with a bubble. (Two stages must be nullified to account for the twice relative speed of the main pipeline.) When the flush ID reaches the start of the pipeline, the flush control logic restarts the pipeline at the instruction *following* the errant instruction. In the event that multiple stages experience errors in the same cycle, all will initiate recovery but only the Razor error closest to writeback (WB) will complete. Earlier recoveries will be flushed by later ones.

Figure 5(b) shows a pipeline timing diagram of a pipelined recovery for an instruction that fails in the EX stage. As in the previous example, the first failed stage computation occurs in the 4th cycle, when the second instruction computes an incorrect result in the EX stage of the pipeline. This error is detected in the 5th cycle, causing a bubble to be propagated out of the MEM stage and initiation of the flush train. The instruction in the EX, ID and IF stages are flushed in the 6th, 7th and 8th cycles, respectively. Finally, the pipeline is restarted after the errant instruction in cycle 9, after which normal pipeline operation resumes.

In the event a panic signal is asserted, all pipeline state is flushed and the pipeline is restarted immediately after the last instruction to writeback. Panic situations complicate the guarantee of forward progress, as the delay in detecting the situation may result in the correct result being overwritten in the Razor shadow latch. Consequently, after experiencing a panic, the supply voltage is reset to a known-safe operating level, and the pipeline is restarted. Once re-tuned, the errant instruction should complete without errors as long as re-tuning is prohibited until after this instruction completes.

A key requirement of the pipeline recovery control is that it not fail under even the worst operating conditions (e.g., low voltage, high temperature and high process variation). This requirement is met through a conservative design approach that validates the timing of the error recovery circuits at the worst-case subcritical voltage.

2.3 Supply Voltage Control

Many of the parameters that affect voltage margin vary over time. Temperature margins will track ambient temperatures and can vary on-die with processing demands. Consequently, to optimize energy conservation it is desirable to introduce a voltage control system into the design. The voltage control system adjusts the supply voltage based on monitored error rates. If the error rate is very low, it could indicate circuit computation is finishing too quickly and voltage should be lowered. Similarly, a low error rate could indicate changes in the ambient environment (e.g., decreasing temperature), giving additional opportunity to lower voltage. Increasing error rates, on the other hand, indicate circuits are not meeting clock period constraints and voltage should be increased. The optimal error rate depends on a number of factors including the energy cost of error recovery and overall performance requirements, but in general it is a small non-zero error rate.

Figure 6 illustrates the Razor voltage control system. The control systems works to maintain a constant error rate of E_{ref}. At regular intervals the error rate of the system is measured by resetting an error counter which is sampled after a fixed period of time. The computed error rate of the sample E_{sample} is then subtracted from the reference error rate to produce the error rate differential E_{diff}. E_{diff} is the input to the voltage control function, which sets the target voltage of the voltage regulator. If E_{diff} is negative the system is experience too many errors, and voltage should be increased. If E_{diff} is positive the error rate is too low and voltage should be lowered. The magnitude of E_{diff} indicates the degree to which the system is "out of tune".

While control of this system may seem simple on the surface, it is complicated by the slow response time of the voltage regulator. Typical commercial voltage regulators can take 10's of microseconds to adjust supply voltage by 100 mV. Consequently, if the controller reacts too fast or too abruptly, the system could become unstable or go into oscillation. Moreover, an overly conservative control function that is slow to react to changing system environments will reduce the overall efficiency of the design. As a starting point, we have implemented a *proportional control system* [15] which adjusts supply voltage in proportion to the sampled E_{diff}. To prevent the control system from over-reacting and potentially placing the system in an unstable state, the error sample rate is roughly equivalent to the minimum voltage step period.

3 Experimental Evaluation

3.1 Razor Pipeline Implementation

The proposed Razor error detection and correction approach was implemented in a 64-bit Alpha processor. The processor was implemented using a simple in-order pipeline consisting of instruction fetch, instruction decode, execute, and memory/writeback with 8 Kbytes of I-cache and D-cache. The implementation details, as well as a die picture, are shown below in Figure 7. The processor was implemented using a 0.18 µm process and is expected to operate at 200 MHz. After careful performance analysis, it was found that only the instruction decode and execute stages were critical at the worst-case voltage and frequency settings and hence required Razor

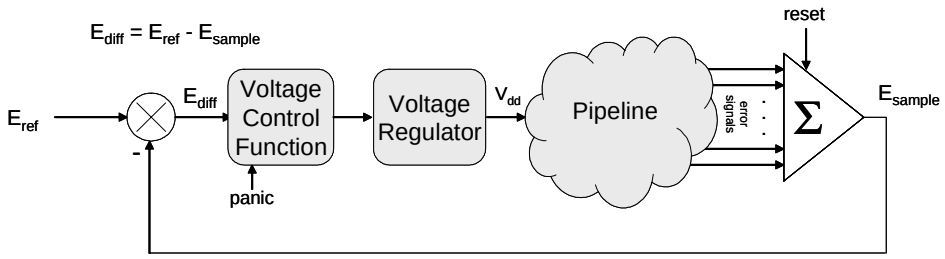
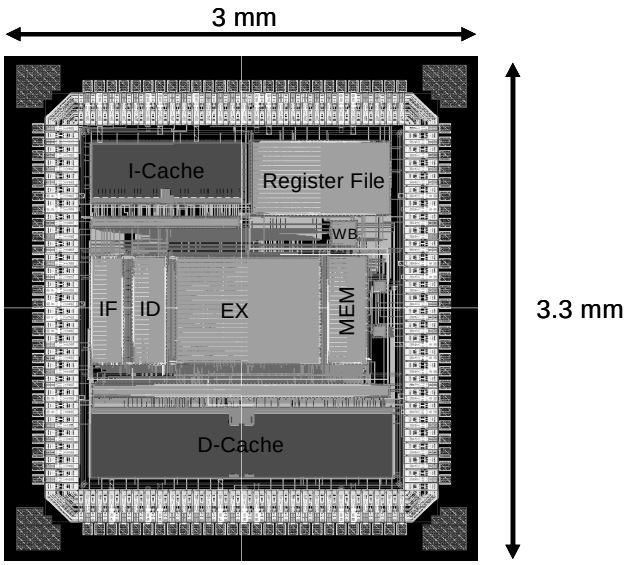


Figure 6. Supply Voltage Control System

Technology node	0.18 µm
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 x 3.3 mm
Clock frequency	200 MHz
Clock delay	2.5 nS
Total number of flip-flops	2408
Number of Razor flip-flops	192
Total number of delay buffers	2498
Error free operation	
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%
Error correction and recovery overhead	
Energy per Razor FF per error event	210 fJ
Total energy per error event	189 pJ
Razor FF recovery overhead at 10% error rate	1%

(a)



(b)

Figure 7. Razor prototype implementation details and die photo.

flip-flops for their critical paths. Out of a total of 2408 flip-flops in the design, 192 Razor flip-flops were used. The clock for the Razor flip-flops was delayed by 1/2 the clock cycle from the system clock.

Power analysis was performed on the processor design, using both gate level power simulations and SPICE to evaluate the overhead of the error correction and detection circuits. The total power consumption during error free operation is expected to be 425 mW at 1.8 V at a clock frequency of 200 MHz. The energy consumption of the standard and Razor flip-flops over one clock cycle in error free operation is listed in Figure 7(a). Two values are shown for each flip-flop, reflecting the cases when the latched data is changing (*switching*) and is not changing (*static*). The total power overhead due to the insertion of delay buffers to meet short-path constraints in the design was simulated and is expect to be 12.2 mW. The total power overhead due to the presence of the Razor error detection and correction circuitry in error-free operation is expected to be 3.1% of the total power. The final three rows of the table show the power overhead due to error detection and recovery. The energy required to detect an error and restore the correct shadow latch data into the main flip-flop was 210 fJ per error event for each Razor flip-flop. The total energy to perform a single error detection and correction event in the Alpha pipeline was 189 pJ, resulting in a power overhead of approximately 1% of total power when operating at a 10% error rate. Note that this error detection and correction power overhead does not include the overhead due to re-execution of instructions that were flushed from the pipeline. This additional power overhead is accounted for in the architectural simulations discussed in Sections 3.4 and 3.5.

3.2 Error rate analysis

Razor permits a microprocessor to tolerate circuit timing errors, thereby permitting operation at a lower voltage at the expense of decreased instruction throughput. As an initial step in gauging the benefits of Razor technology, we empirically examined the error rate of an 18x18-bit multiplier block contained within a high-density FPGA. In addition, we used SPICE-level models to measure the error rates of an adder over a range of voltages and workloads.

FPGA-based analysis. The multiplier experiments were performed using a Xilinx XC2V250-F456-5 FPGA [25]. This part was selected because it contains full-custom 18x18-bit multiplier blocks, which permit the measurement of error rates for a multiplier with minimal impact due to the overhead of the FPGA routing fabric. Figure 8 illustrates the multiplier circuit under test (shaded in the schematic) and accompanying test harness. The multiplier circuit implements an 18-bit by 18-bit multiplier, producing a 36-bit result each clock cycle. During placement, synthesis was directed to foremost optimize the performance of the fast multiplier pipeline. The resulting placement is fairly efficient with the Xilinx static timing

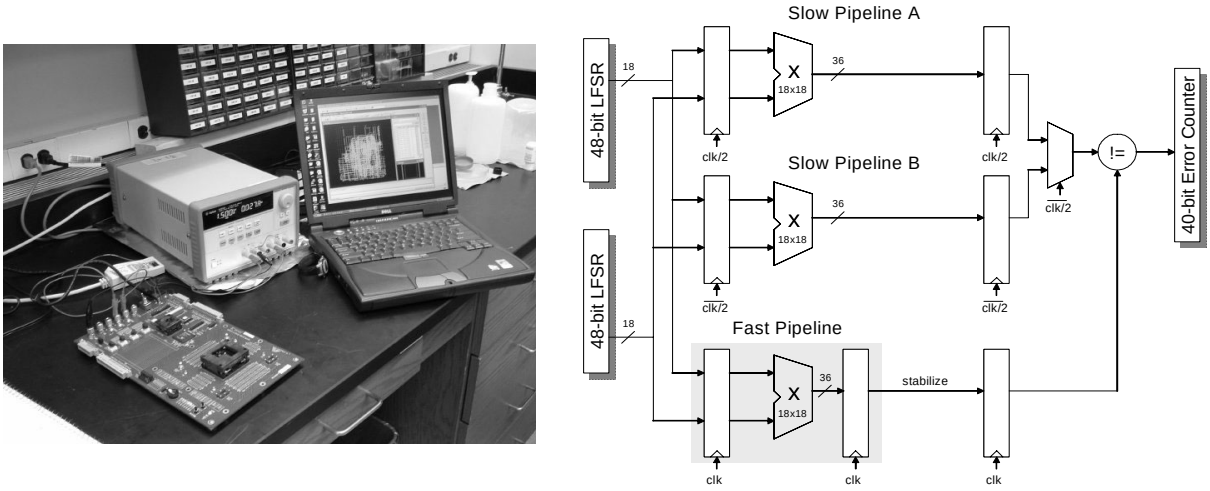


Figure 8. Multiplier Experiment Test Bench and Circuit Under Test.

analyzer (TRCE) indicating that 82% of the fast multiplier stage latency is in the custom multiplier block.

Each cycle, two 48-bit linear feedback shift registers (LFSR) generate 18-bit uncorrelated random values, which are sent to a fast multiplier pipeline, and in alternating cycles to slow multiplier pipelines. The slow multiplier pipelines take turns safely computing the fast pipeline's results, using a clock period that is twice as long as the fast multiplier pipeline. The empty stage after the fast multiplier stage (labeled *stabilize*) allows potentially meta-stable results from the fast multiplier time to stabilize before they are compared with the known-correct slow multiplier results. A MUX on the output of the slow multiplier pipelines selects the correct result to compare against the stabilized output of the fast multiplier pipeline. If the result of the fast pipeline does not match the slow pipeline, an error counter is incremented. The performance of the design was first analyzed with the Xilinx static timing analyzer after back-propagation of FPGA interconnect capacitance. The timing analyzer indicated that the fast multiplier stage could be clocked up to 83.5 MHz at 1.5 V and 85 C. At room temperature 27 C and 1.5 V, the timing analyzer indicated that the design can run at 88.6 MHz. After the fast multiplier, the next longest critical path in the design is the 40-bit error counter, which works up to 140 MHz. As a result, we are confident that all errors experienced in these experiments are localized to the fast multiplier pipeline circuits.

Figure 9 illustrates the relationship between voltage and error rates for an 18x18-bit 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. Also shown on the graph are three additional design points, gauged using the Xilinx static timing analyzer (TRCE). The *zero-margin* point is the lowest voltage where the circuit operates error-free at 27 C. The *safety-margin* point is the voltage at which the circuit runs without errors at 27 C in 90% of the baseline clock period (i.e., 10ns at 100 MHz). We would expect this to be approximately the voltage margin required for delay-chain tuning approaches, where voltage margins are necessary to accommodate intra-die process and temperature variations. Finally, the *environmental-margin* point is minimum voltage required to run without errors at 90% of the baseline clock period at the worst-case operating temperature of 85 C.

As shown in Figure 9, the multiplier circuit fails quite gracefully, taking nearly 200 mV to go from the point of the first error (1.54 V) to an error rate of 5% (1.34 V). Strikingly, at 1.52 V the error rate is approximately one error every 20 seconds, or put another way, 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 those changes in circuit inputs that require a complete re-evaluation of the critical path results in a timing error. As the voltage continues to drop, more and more internal multiplier circuit paths cannot complete within the clock cycle and the error rate increases. Eventually, voltage drops to the point where none of the circuit paths can complete in the clock period, and the error rate reaches 100%. Clearly, if the pipeline can tolerate a small rate of multiplier errors, it can operate with a much lower supply voltage. For instance, at 1.36 V the multiplier would complete 98.7% of all operations without error, for a total energy savings (excluding error recovery) of 22% over the zero-margin point, 30% over the safety-margin point, and 35% over the environmental-margin point.

SPICE-level analysis. To gain a deeper understanding of the nature of circuit timing errors, a circuit-level design of a 64-bit Kogge-Stone adder was implemented and analyzed. A Kogge-Stone adder is a high-performance carry-prefix adder used in a number of commercial microprocessor designs [17]. The Kogge-Stone adder is implemented with the TSMC 0.18 µm standard cell library. The capacitance and resistance for cell interconnect was estimated based on standard cell dimensions and adder topology. The delay of the standard cells were characterized for varied voltages, temperatures and fan-out. A similar delay characterization was performed for interconnect with varied wire lengths. Using these circuit-level characterizations, a high-performance C-level timing model of the Kogge-Stone adder was implemented and validated against SPICE simulations of the same baseline model. We rely on a C-level model to increase the number of sample vectors we can examine, and we integrated this model into an architectural simulator to examine the performance of the adder running with real programs. Comparing the C-model to SPICE simulations (using HSPICE version 2001.2), we found that the error for 50 random vectors never exceeded 10%. Using the C-level models, we then generate error rate estimates using 32,000 sample vector sequences. At a given frequency and voltage, the error rate is computed as the fraction of sample vectors that do not complete within the clock period.

Figure 10 shows the error rate of the Kogge-Stone adder, as a function of voltage, for three 32,000 long input sequence samples. For all experiments, error analysis was performed assuming an 870 MHz clock and an ambient temperature of 27 C. The sample labeled *random* is a random input sequence. The samples labeled *ammp* and *bzip* are adder operations sampled from the SPEC2000 benchmarks with the same name. The benchmark samples were generated by instrumenting the SimpleScalar v3.0 simulator [2] such that all

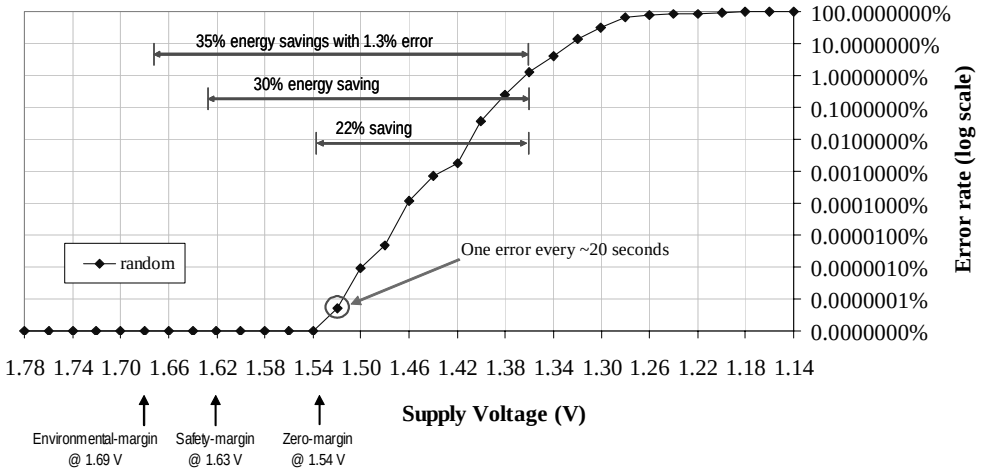


Figure 9. Measured Error Rates for an 18x18-bit FPGA Multiplier Block at 90 MHz and 27 C.

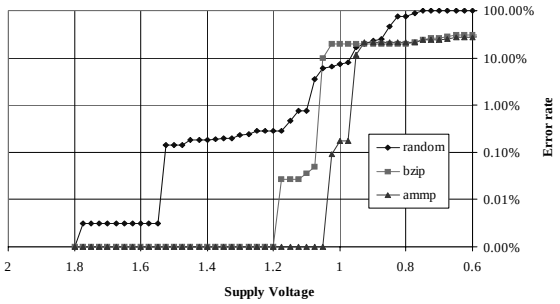


Figure 10. Simulated Error Rates for a Kogge-Stone Adder at 870 MHz and 27 C.

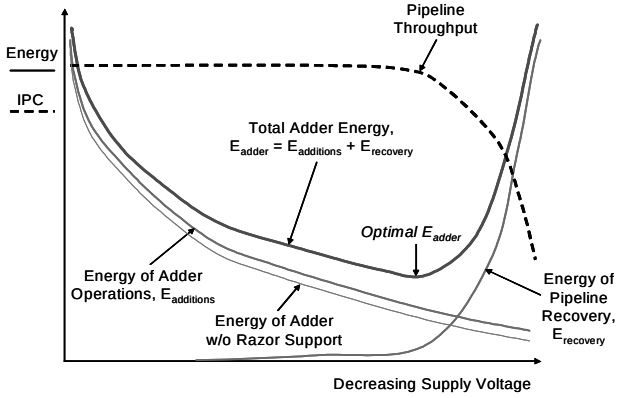


Figure 11. The Qualitative Relationship Between Supply Voltage, Energy and Pipeline Throughput (for a fixed frequency)**.**

instructions using the adder (e.g., adds, subtracts, loads, stores) recorded their inputs. The benchmark samples are taken in program execution order starting at the SimPoint point of the execution, as specified by Sherwood *et al.* [18].

As shown in Figure 10, the random input, like for the multiplier, demonstrates a gradual rise in the error rate with decreasing voltage. We see a similar trend for the benchmark samples analyzed. The error rates for the real program samples increase even more slowly at first than the random sample sequence. For instance, the *ammp* benchmark experiences very few errors until 1.05 V, and *bzip* does not generate any substantial error rates until 1.2 V. With real program samples, the error rate tends to rise faster once errors do take hold, even performing slightly worse than the random sequence at lower voltages. However, at error rates that we would expect to be easily tolerated (e.g., below 5%), the real program samples demonstrate substantially lower operating voltages than the random sample sequence.

3.3 Simulator Framework and Benchmarks

The architectural simulators used in this paper are derived from the SimpleScalar/Alpha version 3.0 tool set [2], a suite of functional and timing simulation tools for the Alpha AXP ISA. Simulation is execution-driven, including execution down any speculative path until the detection of a fault, TLB miss, or branch misprediction. The baseline processor modeled was a single-issue, in-order pipeline with the pipeline stages that are described in Section 3.1. The baseline model was modified to simulate Razor error recovery with its proper penalties. Furthermore, the detailed C-level Kogge-Stone adder model was integrated into the execute stage, where it was used to determine when voltage scaling introduced adder timing errors.

To perform our evaluation, we collected results from 11 of the SPEC2000 benchmarks. All SPEC programs were compiled for a Compaq Alpha AXP-21264 processor using the Compaq C and Fortran compilers under the OSF/1 V4.0 operating system using full compiler optimization (-O4). The simulations were run for 100 million instructions using the SPEC reference inputs. We used the SimPoint toolset's Early SimPoints to pinpoint program locations that were highly representative of the entire program execution [18].

3.4 Energy Analysis for Fixed Voltage

Figure 11 illustrates *qualitatively* the relationship between supply voltage, adder energy and pipeline throughput. The total energy consumed by the adder (E_{adder}) is the sum of the energy required to perform add operations ($E_{additions}$) plus the energy required to recover the pipeline in the event of an adder timing error ($E_{recovery}$). Moreover, there is a fixed amount of energy overhead incurred to implement Razor checking for the adder. This energy is consumed by the shadow latches and comparison logic. A trade-off exists between the adder and recovery energy components. When supply voltage is decreased, the energy required to perform addition operations is decreased, but fewer of these operations are able to complete within the clock period. As a result, pipeline recovery is invoked more frequently with additional energy expense. Energy for the adder (E_{adder}) is optimized when any additional decrease in voltage

Table 1. Energy-Optimal Characteristics

Program	Optimal V_{dd}	Error Rate	% Energy Reduced	% IPC Reduced
bzip	1.1	0.31%	57.6%	0.70%
crafty	1.175	0.41%	50.5%	0.60%
eon	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%*	

results in an energy savings that is smaller than the extra energy cost incurred by more pipeline recoveries. The energy-optimal voltage varies from program to program (and even within the phases of a program) because pipeline error rate is heavily dependent on the data values sent to the adder. These trade-offs are further complicated under a pipeline performance constraint. Decreasing voltage will incur additional pipeline errors, which in turn decreases pipeline throughput (i.e., instructions per cycle). Consequently, the program will take longer to execute. Under a performance constraint, the optimal voltage is limited to the minimal energy that meets the performance constraint.

Table 1 lists for each benchmark the energy-optimal supply voltage, average adder error rate, energy reduction, and IPC reduction at the fixed energy-optimal voltage. The simulations are performed by sweeping the voltage in 25 mV steps from 1.8 V down to 0.6 V. The voltage remains fixed for the entire simulation (i.e., each point on the graph is a different simulation). All experiments are performed at 27 C and 870 MHz, the maximum speed at which the adder runs error-free at room temperature (i.e., the zero-margin point). All Razor energy estimates were made using RTL-level power analysis of the Razor prototype physical design described in Section 3.1. The total energy of the Razor adder includes the energy of the adder, Razor latch and check circuitry, and the total pipeline recovery energy incurred when a Razor adder error is detected. The Razor latches and error detection circuitry increase adder energy by about 4.3%. Error recovery energy is conservatively estimated at 18 times the cost of a single add (at 1.8 V), based on a 6-cycle recovery sequence at typical activity rates. It should be noted that the energy savings reflect only that due to eliminating data-dependent delay margins. If comparisons were made to existing DVS techniques that require safety margins (e.g., delay line speed detector) or temperature margins (e.g., design-time DVS), the resulting energy saving would be substantially higher. Table 1 also shows the relative performance of the benchmark, given as the IPC of the program with Razor timing speculation divided by the IPC of a non-speculative pipeline. Since all the experiments are run at the same frequency, the change in IPC due to pipeline recovery reflects true performance impacts. Figure 12 illustrates the relative energy and performance across the entire supply voltage operating range, for the benchmarks *bzip* and *gcc*.

It is important to note that the energy analysis presented in this section only reflects the energy savings in the Razor pipeline adder.

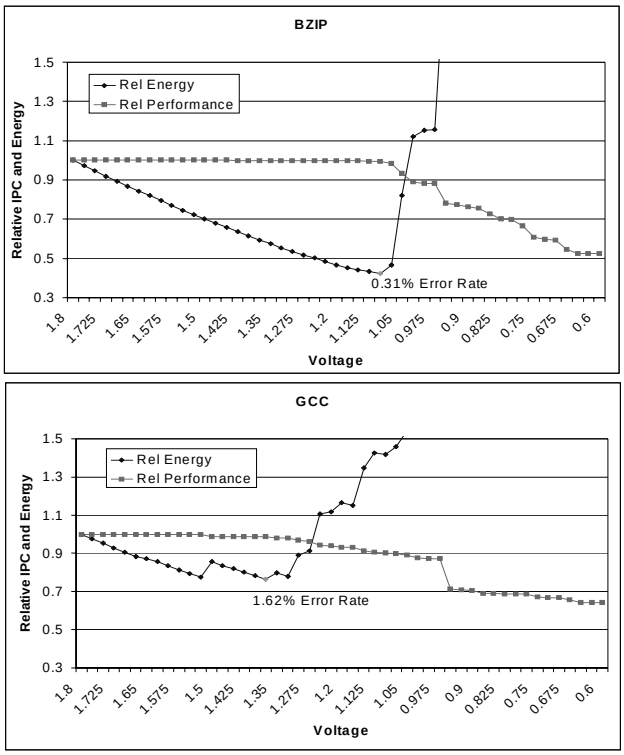


Figure 12. Relative Adder Energy and Pipeline Throughput for Simulated Benchmarks.

We only consider the energy of the entire Razor pipeline when an adder timing error occurs. In this event, all activities (and pipeline energy) are directly attributable to the Razor timing error, and thus must be counted against Razor adder energy savings. In essence, this is the adder energy savings one could expect if the adder were given its own independently tunable voltage source. Total energy reduction for the entire pipeline would only be the same if the remaining components could scale their voltage to the same degree without increasing the overall error rate.

Clearly, there is significant energy to be reclaimed by running the adder at a low error rate. All of the benchmarks experienced significant energy savings, ranging from 23.7% to 64.2%. One particularly encouraging result is that error rates and performance impacts are muted up to and slightly past the energy-optimal voltage, after which error rates rise very quickly. At the energy-optimal voltage point, the benchmarks suffered at most a 2.49% reduction in pipeline performance (due to recovery flushes). There appears to be little trade-off in performance when fully exploiting adder energy savings at subcritical voltages. While we have simulated voltages down to 0.6V, our Razor prototype design is only capable of validating 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 on future prototype designs.

3.5 Energy Analysis for Dynamic Voltage Scaling

Reducing voltage to the energy-optimal fixed voltage point will certainly improve the energy characteristics of a system that employs Razor. In this section, we consider the potential value of dynamically adjusting supply voltage to workload characteristics. We perform these experiments by engaging the proportional control system described in Section 2.3. For the simulated experiments, we assume a voltage regulator response time of 20 cycles per 1 mV. The control system samples (and then resets) an error counter every 5000 cycles, and adjusts the voltage regular plus or minus 25 mV, depend-

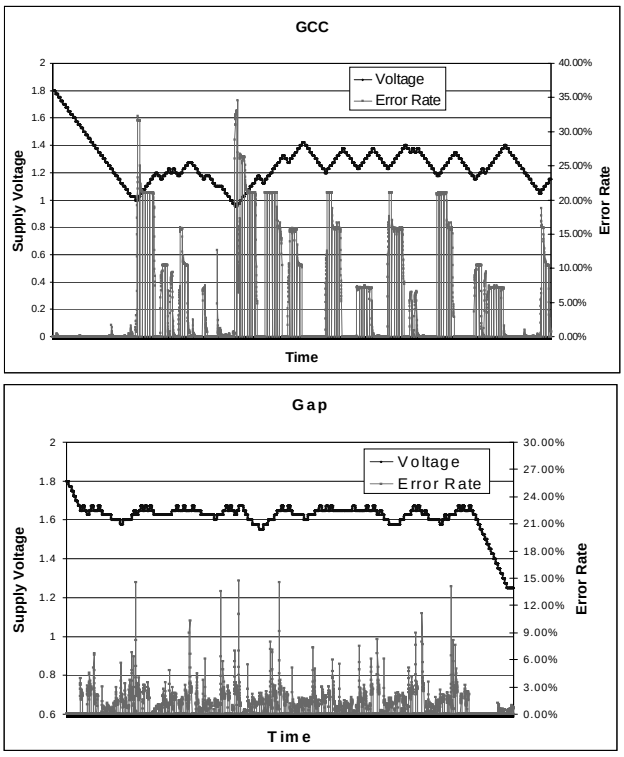


Figure 13. Adder Error Rate and Voltage Controller Response.

ing on the error rate differential. All simulations use a target error rate of 1.5%, which was set based on the energy-optimal error rates analyzed in the previous section.

Table 2 lists the adder energy reduction compared to a non-Razor adder at the zero-margin voltage (1.8 V). Dynamically adjusting voltage again results in substantial energy savings. Compared the fixed voltage experiments of the previous section, about one half of the benchmarks see better energy savings, and the other half has slightly worse energy savings. With dynamic voltage scaling, most of the benchmarks ran slower, although overall performance impacts were still small, with the largest slowdown limited to just under 6%. Figure 13 illustrates the change in error rate of the adder over time and the voltage control systems response to the error rate for the *gcc* and *gap* benchmarks. Overall, the results for the proportional control system are mixed. Given that it represents a fairly unsophisticated class of control functions; further investigations into supply voltage control will likely yield additional energy savings.

4 Previous Work

Table 3 lists a number of prior proposals supporting adaptive voltage and frequency scaling. With *Design-Time DVS*, conservative design techniques are used to specify "legal" voltage and frequency pairs that allow reliable operation of the processor under worst-case voltage, temperature, and process conditions. Examples of systems that utilize this approach are Intel's x86 SpeedStep technology [10] and Transmeta's Longrun technology [20].

A *Correlating VCO* allows ambient margins to be eliminated; examples of this design have been proposed by Burd [3] and Gutnik [7]. The approach implements a voltage controlled oscillator using a timing loop constructed to slightly exceed the latency of the worst-case critical path in the machine, plus process and safety margins. When supply voltage changes, the oscillator speed will automatically adjust to match the fastest safe clock speed. It is important to note that this approach cannot compensate for intra-die process and temperature variations, IR drop, or noise. As a result, additional voltage margins (implemented with additional timing loop delay) are required for safe operation.

Table 2. Simulated DVS Energy Savings

Program	% Energy Reduced	% IPC Reduced
bzip	54.5%	4.13%
crafty	54.8%	1.78%
eon	30.4%	0.78%
gap	12.9%	2.14%
gcc	31.3%	5.88%
gzip	44.6%	1.27%
mcf	36.9%	0.47%
parser	53.0%	1.94%
twolf	20.4%	0.06%
vortex	49.1%	1.07%
vpr	63.6%	1.66%
Average	*41.0%*	

A *Delay Line Speed Detector* is a device that models the worst-case critical path of the system, plus a safety margin. Examples of these devices have been proposed by Dhar [5] and Uht [21]. Periodically, a signal transition is propagated down a delay chain and sampled at the end of the current clock cycle. If the signal transition does not propagate to the end of the delay chain within the clock period, the system is running too close to failure and frequency and/or voltage must be adjusted. Since the delay chain fails prior to the core circuitry, any failure detected in the delay chain will proceed a core circuitry failing, assuming that the delay line is frequently monitored and the system is adjusted promptly upon detection of a delay line failure. To ensure that the delay line fails first, it is necessary to add latency margins to accommodate intra-die process and temperature variations, IR drop and noise. Unlike the Correlating VCO, it may be possible to put multiple delay line speed detectors across the die and combine their timing signals in an effort to mitigate intra-die process and temperature variation. However, some variation is inherently local (e.g., cross-coupling noise), thus some delay margin will always be required. We have not seen the use of multiple delay line detectors explored to date.

Kehl's *Triple-Latch Monitor* is similar to the delay line speed detector, but like Razor, utilizes *in-situ* circuit monitoring [11]. Using this approach, all monitored system state is captured using three latches, clocked in succession with a small delay between each. The staggered latches provide three closely spaced samples of a logic block's value each cycle. The value in the latest-clocked latch is assumed correct and always forwarded to later logic. The system is considered "tuned" when the first latch does not match the second and third latch values, meaning that the logic transition was very near the critical speed, but not dangerously close. If all latches see the same value, the system is running too slowly and should be sped up. If the first two latches see different values than the last, then the system is running dangerously fast and should be slowed down. Because of the *in-situ* nature of this approach, it could conceivably adjust to intra-die process and temperature variations. However, data-dependent delay variations complicate Kehl's approach. To avoid too aggressively clocking the system, speedup evaluations must be limited to tests on worst-cast latency vectors. Kehl suggests that the system should periodically stop and test worst-case vectors to determine if the system should be sped up.

Table 3. Adaptive Voltage/Frequency Scaling Landscape.

	Margin Eliminated?				
Technique	**Data**	**Process**	**Environmental**	**Safety**	**Speculative?**
Design-time DVS [10][20]	N	N	N	N	N
Correlating VCO [3][7]	N	N	Y	N	N
Delay Line Speed Detector [5][21]	N	N	Y	N	N
Triple-Latch Monitor [11]	N	Y	Y	N	N
Circuit-Level Speculation [12][24]	Y	N	N	N	Y
Razor	Y	Y	Y	Y	Y

Circuit-level Speculation employs logic components that operate at two speeds, a fast typical speed and a slower atypical multi-cycle speed. The components are designed with typical usage in mind, which in all published cases resulted in significantly favorable circuit speed due to shorter data-dependent circuit paths. Two prior proposal of this nature include Liu's fast adder and scheduler designs [12] and Wolrich's stutter adder [24]. Both fast adder designs were optimized to perform short-distance carry propagation in a single cycle, with longer carry propagations taking an additional cycle. Liu's circuit-speculative scheduler provided very fast access to a few instructions. If dependencies warranted wake-up of other instructions, multiple cycles were required. Like Razor, circuit-level speculation benefits by exploiting typical-case evaluation latency, which for most workloads is much more favorable than worst-case latency. Unlike Razor however, circuit-level speculation cannot adapt timing to changing workload or other margin factors such as temperature or process variation. Moreover, it is unclear how circuit-level timing speculation could be adapted to dynamic voltage scaling.

We are aware of three previous proposals that suggest using rate-matched redundant hardware to allow subcritical circuit operation. Uht's TIMERRTOL design methodology couples an overclocked logic block with multiple safely clocked blocks of the same logic [22]. By using multiple check logic blocks, his approach can check all overclocked computation with hardware blocks that are safely clocked. Uht does not address the possibility of metastability in the fast block's output latches or the problem of recovering system state after a timing error. Razor addresses both of these issues and utilizes an implementation that is much less expensive. Austin suggested that the DIVA checker could be over-designed to validate computation from an overclocked core processor [1], but the details of how this might be implemented were not explored. Hegde and Shanbhag proposed the use of algorithmic noise tolerance (ANT) to permit the operation of signal processing circuits at subcritical voltages [9]. They couple the signal processor with a rate-matched error predictor that limits the additional noise incurred by errant circuit computations. Using their approach, voltage can be lowered to the extent that the application can tolerate additional noise in the signal processor output.

Our pipeline recovery mechanism is inspired from Sproull's work on asynchronous counterflow pipelines [19], which was later adapted for synchronous systems by Miller [13]. The basic idea of a counterflow pipeline is that instruction and control signals flow in a direction opposite to data values. As such, global control is not necessary as all control signals will eventually reach the appropriate point in the datapath. We use a counterflow-style pipeline to implement low-complexity recovery of the Razor pipeline in the event of a circuit error.

Razor shares many of the benefits of asynchronous designs, while mitigating many of their drawbacks. Asynchronous systems eliminate the global clock and instead utilize data-driven control to orchestrate system state changes [8],[23]. The approach has long been held up as a promising technique to improve system throughput and power. For example, asynchronous designs readily adapt to data-dependence, ambient and process variation. Unfortunately, the technique is not without drawbacks, including substantial additional design complexity to deal with hazards and ordering of operations, and more complicated system testing. While fundamentally a synchronous system, Razor can also adapt to data-dependence, ambient and process variation. Unlike asynchronous designs, Razor utilizes a traditional synchronous design style using standard tools. An additional detractor for the use of asynchronous logic is its non-deterministic operation. Temperature variation, for instance, can change the order of logic evaluation and state transitions, making functional and electrical validation more challenging. While Razor shares this non-determinism, we feel it will not put undue burden on the verification process for two reasons. First, non-determinism is limited to whether or not a stage of the pipeline will produce an error. Bugs relating to the non-deterministic nature of the Razor pipeline will be confined to the error recovery machinery. Second, it should be possible to provide verification-time buffering of stage error signals, which would permit deterministic replay of non-deterministic executions. This support would address any reproducibility concerns during verification.

5 Conclusions

In this paper, we presented Razor, an error-tolerant dynamic voltage scaling technology. The key advantage of Razor over existing voltage scaling technologies is the use of *in-situ* timing error detection and correction, permitting increased energy reduction because voltage margins are completely eliminated. The Razor flip-flop was introduced as a mechanism to double-sample pipeline stage values, once with an aggressive fast clock and again with a delayed clock that guarantees a reliable second sample. A metastability-tolerant error detection circuit was described that validates all values latched on the fast Razor clock. In the event of a timing error, a modified pipeline flush mechanism restores the correct stage value into the pipeline, flushes earlier instructions, and restarts the next instruction after the errant computation.

A prototype Razor pipeline was designed and analyzed. We found that during normal (error-free) operation of the pipeline, Razor error detection increases pipeline energy demands by a modest 3.1%, compared to a non-Razor design of the architecture. Energy requirements for error recovery were much greater. We found that the energy required to fully recover the pipeline after an adder timing error was about 18 times more expensive than the errant addition.

The error rates of real and simulated circuits were explored in detail. A full-custom 18x18-bit FPGA multiplier block confirmed

that significant energy reductions are possible for real circuits, if small error rates can be tolerated. When computing on random inputs at room temperature, the multiplier circuit consumed 17% less energy when all process and temperature margins on voltage were eliminated. Continuing to decrease voltage to the point where 1.3% of operations fail consumes 35% less energy. Detailed analysis of a SPICE-level Kogge-Stone adder model reveals that real program data has more favorable error rates than random samples. Compared to random inputs, real program inputs see similar error rates at a voltage that is nearly 400 mV lower.

Architectural simulations were performed to gauge the benefits of Razor DVS in the presence of potentially expensive pipeline recoveries. Simulations at the fixed energy-optimal voltage for each benchmark revealed that even with high pipeline recovery costs (in terms of energy and performance) a Razor adder operated with 42% less energy, while only incurring at most a 2.5% reduction in pipeline throughput. The introduction of a proportional voltage control system performed nearly as well overall, suggesting that near energy-optimal voltage points could be found automatically for individual program. In some cases, the voltage control system performed better than running with a fixed energy-optimal voltage, suggesting that program energy demands are phasic. It is likely that further improvement to the voltage control system would render additional savings.

Looking ahead, there is much more ground to explore. In mid-November 2003, we tape-out our prototype Razor pipeline design for MOSIS fabrication. A few months later, we will have the first opportunity to analyze a complete Razor pipeline design. To increase the scope of Razor, we have begun exploring its application to memory structures and pipeline control logic. Finally, there is a great opportunity to "re-think" system design in the context of Razor. In particular, we want to investigate the design of functional units and memory structures optimized for typical-case latency. These new designs should have lower error rates, thereby creating additional opportunity to lower energy demands.

Acknowledgements

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

References

[1] T. Austin, "DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design," *32nd Int'l Symposium on Microarchitecture*, Nov. 1999.
[2] T. Austin, E. Larson, D. Ernst. SimpleScalar: an Infrastructure for Computer System Modeling, *IEEE Computer*, 35 (2), February 2002.
[3] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *Int'l Solid-State Circuits Conf.*, Feb. 2000.
[4] W. Dally, J. Poulton, *Digital System Engineering*, Cambridge Press, 1998
[5] S. Dhar, D. Maksimovic, and B. Kranzen, "Closed-Loop Adaptive Voltage Scaling Controller For Standard-Cell ASICs," *2002 Int'l Symposium on Low Power Electronics and Design (ISLPED-2002)*, August 2002.
[6] R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS," IEEE JSSC, 32 (8), August 1997.
[7] V. Gutnik and A. Chandrakasan, "An Efficient Controller for Variable Supply-Voltage Low Power Processing," *Symp. on VLSI Circuits*, June 1996.
[8] S. Hauck, "Asynchronous Design Methodologies: An Overview," *Proceedings of the IEEE*, 83 (1), January 1995.
[9] R. Hegde and N. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," *1999 International Symposium on Low-Power Electronics and Design (ISLPED-99)*, August 1999.
[10] Intel Corp., "Intel SpeedStep Technology," http://www.intel.com.
[11] T. Kehl, "Hardware Self-Tuning and Circuit Performance Monitoring," *1993 Int'l Conference on Computer Design (ICCD-93)*, October 1993.
[12] T. Liu and S. Lu, "Performance Improvement with Circuit-Level Speculation," *33rd Annual International Symposium on Microarchitecture (MICRO-33)*, December 2000.
[13] M. Miller, K. Janik and S.-L. Lu, "Non-Stalling Counterflow Microarchitecture," *4th International Symposium on High Performance Computer Architecture (HPCA-4)*, February 1998.
[14] T. Mudge. "Power: A first class design constraint," *Computer*, vol. 34, no. 4, April 2001, pp. 52-57.
[15] K. Ogata, "Modern Control Engineering," 4th ed., Prentice Hall, 2002.
[16] T. Pering, T. Burd, and R. Brodersen. "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms." *Proceedings of Int'l Symposium on Low Power Electronics and Design 1998*, pp. 76-81, June 1998.
[17] J. Rabaey, "Digital Integrated Circuits," Prentice Hall, 1996.
[18] T. Sherwood, E. Perelman, G. Hamerly and B. Calder, "Automatically Characterizing Large Scale Program Behavior," 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), October 2002.
[19] R. Sproull, I. Sutherland, and C. Molnar, "Counterflow Pipeline Processor Architecture," *Sun Microsystems Report SMLI-TR-94-25*, April 1994.
[20] Transmeta Corporation, "LongRun Power Management," http://www.transmeta.com/technology/architecture/longrun.html.
[21] A. Uht, "Uniprocessor Performance Enhancement Through Adaptive Clock Frequency Control," *2003 International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, e-Medicine, and Mobile Technologies on the Internet (SSGRR 2003w)*, January 2003.
[22] A. Uht, "Achieving Typical Delays in Synchronous Systems via Timing Error Toleration," *University of Rhode Island TR-032000-0100*, March 2000.
[23] S. Unger, "Asynchronous Sequential Switching Circuits," New York: Wiley-Interscience, John Wiley & Sons, Inc., 1969.
[24] G. Wolrich, E. McLellan, L. Harada, J. Montanaro, and R. Yodlowski, "A High Performance Floating Point Coprocessor," IEEE Journal of Solid-State Circuits, 19 (5), October 1984.
[25] Xilinx Corporation, "Virtex-II Platform FPGA," http://www.xilinx.com/products/tables/fpga.htm#v2