Chapter 8 Architectural Techniques for Adaptive Computing

[1,2]Shidhartha Das, [2]David Roberts, [2]David Blaauw, [1]David Bull, [2]Trevor Mudge

[1]ARM Ltd., UK, [2] University of Michigan

8.1 Introduction

As critical geometries shrink to the 45nm region and beyond, lithographic limitations have led to rising intra- and inter-die process variations. Increased variability makes it significantly difficult to accurately model transistor behavior on silicon, and often probabilistic methods are required [1]. The consequent loss in silicon predictability implies that design uncertainties become severe and are made even worse at the lower supply voltages used for future technologies [2].

In addition to process variability, deep sub-micron technologies also suffer from increased power consumption which compromises structural reliability of processors. Indeed, as current densities have increased, chip failure through effects like electro-migration [3] and time-dependent dielectric breakdown (TDDB) [4] has become major challenge, especially for high-end processors. Furthermore, at lower supply voltages, noise margins for sensitive circuits significantly reduce. Consequently, signal integrity concerns assume greater relevance. Smaller noise margins enhance susceptibility to capacitive and inductive coupling, thereby adversely affecting computational robustness. Robustness is further aggravated by resistive voltage drops and inductive overshoots in the supply voltage network. As such, it will be exceedingly difficult to sustain the current rate of technology scaling unless power and robustness concerns are suitable addressed [5].

A. Wang, S. Naffziger (eds.), *Adaptive Techniques for Dynamic Processor Optimization*,
DOI: 10.1007/978-0-387-76472-6_8,

The traditional approach of fabricating robust circuits has been to design for the worst-case scenario. In this approach, circuits are built with sufficient safety margins such that they operate correctly even under the worst-case combination of process, voltage and temperature conditions. As design uncertainties worsen, it is expected that safety margins will increase at future technology nodes. At these nodes, the worst-case transistor performance is likely to vary widely from that under typical conditions. This limits the operating frequency of processors, thereby reducing the performance improvements that technology scaling traditionally afforded. Furthermore, safety margins typically require the use of wider devices, higher operating voltage and thicker interconnects, all of which have the undesirable effect of increased power consumption. Thus, while design margining ensures robust operation, unfortunately, it also leads to reduced performance and increased power consumption.

A key observation is that robust computing and low power are fundamentally at odds with each other. Low-power methodologies typically sacrifice robustness for lower power consumption and vice versa. This trade-off is especially significant in the mobile and battery-operated world where meeting robustness and performance targets under restrictive power budgets makes design closure difficult. For example, an effective low-power technique is dynamic voltage scaling (DVS), which enables quadratic power savings by scaling supply voltage during low CPU utilization periods. However, low voltage operation causes signal integrity concerns by reducing the static noise margins for sensitive circuits. Furthermore, sensitivity to threshold voltage variation also increases at low voltages [2] which can lead to circuit failure. Another popular technique for low power relies on downsizing off-critical paths [6]. This balances path delays in the design leading to the so-called *timing wall*. In a delay-balanced design, the likelihood of chip failure significantly increases because more paths can now fail setup requirements. Conversely, most robust design techniques, such as hardware redundancy and conservative margining, hurt power consumption. Thus, the traditional design paradigm leads to a very complex optimization space where design closure by simultaneously meeting power, performance, and robustness objectives can be exceedingly difficult.

In order to effectively address the issue of design closure, it is helpful to analyze and categorize the sources of design uncertainties, depending on their spatial reach and temporal rate of change.

8.1.1 Spatial Reach

Based on spatial reach, design uncertainties can be further subdivided as follows:

- ***Global uncertainties***

Those that affect all transistors on the die are *global* in nature. For example, global supply voltage variations affect the entire die and could be due to voltage fluctuations onboard or within the package. Other examples of such global phenomena are inter-die process variations and ambient temperature.

- ***Local uncertainties***

Local effects are limited to a few transistors in the immediate vicinity of each other. Voltage variations due to resistive drops in the power grid and temperature hot spots in regions of high switching activity have local effects. Cross-coupling noise events are extremely local and are restricted to a few signal nets near the aggressor. Other examples of local effects are intra-die process variations.

8.1.2 Temporal Rate of Change

Based on their rate of change with time, design uncertainties can be broadly divided under the following categories.

- ***Slow-changing effects***

Design uncertainties that have time constants of the order of millions of cycles or more can be categorized as slow-changing. Thus, they could be

(**a) Invariant with time:** Effects such as intra- and inter-die process variations are fixed after fabrication and remain effectively invariant over the lifetime of the processor.

(b) Extremely slow-changing, spread over the lifetime of the die: Wear-out mechanisms such as negative bias temperature instability [7], TDDB [4] and electro-migration are typical examples of such effects that gradually degrade processor performance over its lifetime.

(c) Moderately slow-changing, spread over millions of cycles: Temperature fluctuations fall under this category.

- ***Fast-changing effects***

Such effects develop over thousands of cycles or less. They could be

(a) Moderately fast-changing, spread over thousands of cycles: Supply voltage uncertainties attributed to the Voltage Regulation Module or

board-level parasitics can cause supply voltage variations on-die. Such effects develop over a range of few microseconds or thousands of processor cycles.

(b) Fast-changing, spread over tens of cycles: Inductive overshoots due to package inductance can cause supply voltage noise with time constants of the order of tens of processor cycles.

(c) Extremely fast-changing, spread over a few cycles or less: IR drops in the on-chip power supply network develop over a few cycles. Coupling noise effects exist for even shorter durations; typically for less than a cycle.

In addition to process and silicon conditions, input vector dependence of circuit delay is another major source of variation which cannot be captured easily in the above categories. Circuits exhibit worst-case delay for very specific instruction and data sequences [8]. Consequently, most input vectors do not sensitize the critical path, thereby aggravating the pessimism due to overly conservative safety margins.

Addressing the issue of excessive margins requires a fundamental departure from the traditional technique of operating every dice at a single, statically determined operating point. Adaptive design techniques seek to mitigate excessive margining by dynamically adjusting system parameters (voltage and frequency) to account for variations in environmental conditions and silicon grade. Thus, a significant portion of worst-case safety margins is eliminated leading to improved energy efficiency and performance over traditional methods. Broadly speaking, adaptive techniques can be divided into two main categories.

- **"Always-correct" techniques**

The key idea of "always-correct" techniques is to predict the point of failure for a die and to tune system parameters to operate near this predicted point. Typically, safety margins are added to the predicted failure point to guarantee computational correctness.

- **"Error detection and correction" techniques**

Such approaches rely on scaling system parameters to the point of failure. Computation correctness is ensured by detecting timing errors and suitably recovering from them.

Table 8.1 compiles a list of different adaptive design techniques discussed in literature and the margins eliminated by each of them. We survey these techniques in detail in Sections 8.2 and 8.3, respectively. In Section 8.4, we discuss "Razor" as a special case study of error detection and correction approaches. In this section, we introduce the basic concepts of Razor. We follow it with measurement results on a test chip using Razor for adaptive voltage control in Section 8.5. Section 8.6 deals with the recent research

related to Razor. Finally, Section 8.7 concludes the chapter with few remarks on the future direction of research on adaptive techniques.

Table 8.1 Adaptive techniques landscape.

Category	Technique	Margins eliminated							General-purpose computing?
		Data	Process		Ambient (V,T)				
			Intra-die	Inter-die	Local		Global		
					Fast	Slow	Fast	Slow	
Always correct	Table look-up [Section 8.2.1]	N	N	N	N	N	N	N	Y
	Canary circuits [Section 8.2.2]	N	N	Y	N	N	N	Y	Y
	In situ triple-latch monitor [Section 8.2.3]	N	Y	Y	N	Y	N	Y	Y
	Typical delay adder structures [Section 8.2.4]	Y	N	N	N	N	N	N	Y
	Non-uniform cache architectures [Section 8.2.4]	Y	N	N	N	N	N	N	Y
Error detection and correction	Self-calibrating interconnects [Section 8.3.1]	Y	Y	Y	Y	Y	Y	Y	N
	ANT [Section 8.3.1]	Y	Y	Y	Y	Y	Y	Y	N
	Razor [Section 8.4]	Y	Y	Y	Y	Y	Y	Y	Y

8.2 "Always-Correct" Techniques

As mentioned before, "always-correct" techniques predict the operational point where the critical path fails to meet timing and to guarantee correctness by adding safety margins to the predicted failure point. The conventional approach toward predicting this point of failure is to use either a look-up table or the so-called *canary* circuits.

8.2.1 Look-up Table-Based Approach

In the look-up table-based approach [9][10][11], the maximum obtainable frequency of the processor is characterized for a given supply voltage. The voltage–frequency pairs are obtained by performing traditional timing

analysis on the processor. Typically, the operating frequency is decided based on the deadline under which a given task needs to be completed. Accordingly, the supply voltage corresponding to the frequency requirement is "dialed in". The table look-up approach is able to exploit periods of low CPU utilization by dynamically scaling voltage and frequency, thereby leading to energy savings. Furthermore, owing to its relative simplicity, this approach can be easily deployed in the field. However, its reliance on conventional timing analysis performed at the combination of worst-case process, voltage and temperature corners implies that none of the safety margins due to uncertainties are eliminated.

8.2.2 Canary Circuits-Based Approach

An alternative approach relies on the use of the so-called canary circuits to predict the failure point [12], [13]–[17], [38]. Canary circuits are typically implemented as inverter chains which approximate the critical path of the processor. They are designed to track the critical-path delay across process, voltage and temperature corners. Voltage and frequency are scaled to the extent that this replica-delay path fails to meet timing. The key requirement for this approach is that the main processor operates correctly even when the replica-delay path fails to meet timing. In order to ensure this, worst-case safety margins are added to the replica path to account for *local* variations due to temperature hot spots, cross-talk noise, power supply droops and intra-die process variation. Furthermore, the replica path cannot respond to the *fast-changing* effects for which worst-case safety margins need to be budgeted. Margins also need to account for mismatches in the scaling characteristics of the replica path and the critical path.

There are several systems reported in literature based on canary circuits. One approach uses the replica path as a delay reference for a voltage-controlled oscillator (VCO) unit. The VCO monitors the delay through the chain at a given supply voltage and scales the operating frequency to the point of failure of the replica path. An example of such an approach is Uht's TEATime which is illustrated in Figure 8.1.

A toggle flip-flop initiates a new transition through the replica path every cycle. The transition is correctly captured at the receiving flip-flop only if the clock period is greater than the propagation delay through the replica path. A simple up–down counter is used to control the VCO frequency output via a digital-to-analog converter (DAC). IBM's PowerPC System-on-Chip design reported in [14] and the Berkeley Wireless Research Center's [16][12] low-power microprocessor are all based on a similar concept. An alternative approach developed by Sony and reported

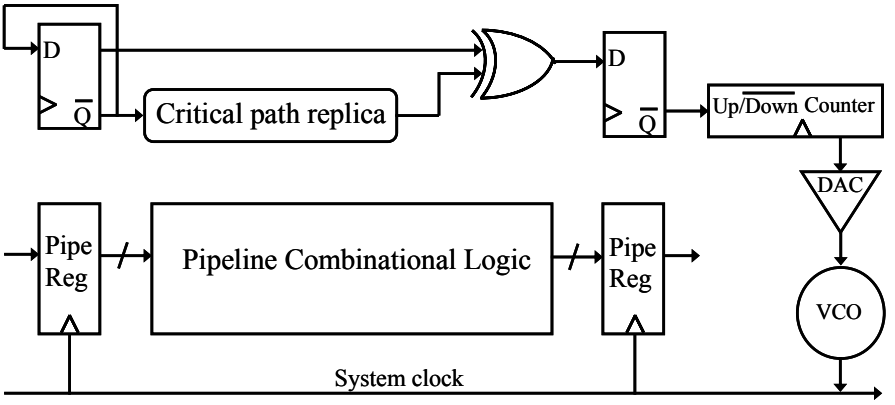


Figure 8.1 Uht's TEATime: A canary circuit-based approach.

in [17] uses a delay synthesizer unit consisting of several delay chains which selects a safe frequency depending on the maximum propagation delay through the chains. Typically, canary circuits enable better energy efficiency than the table look-up approach because unlike the latter, they are able to eliminate margins due to inter-die process variations and *global* fluctuations in voltage and temperature that are essentially *slow-changing* effects [18].

8.2.3 In situ Triple-Latch Monitor

Kehl's triple-latch monitor is similar to the canary circuits-based techniques, but utilizes in situ monitoring of circuit delay [19]. Using this approach, all monitored system states are sampled at three different latches with a small delay interval between each sampling point, as shown in Figure 8.2(a). The value in the latest-clocked latch which is allowed the most time is assumed correct and is always forwarded to later logic. The system is considered "tuned" (Figure 8.2b) 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 frequency should be increased. 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 can adjust to *local* variations such as intra-die process and temperature variations. However, it still cannot track *fast-changing* conditions such as cross-coupling and voltage noise events. Hence, the delay between the successive samples has to be sufficiently separated to allow for margins for such events. In addition,

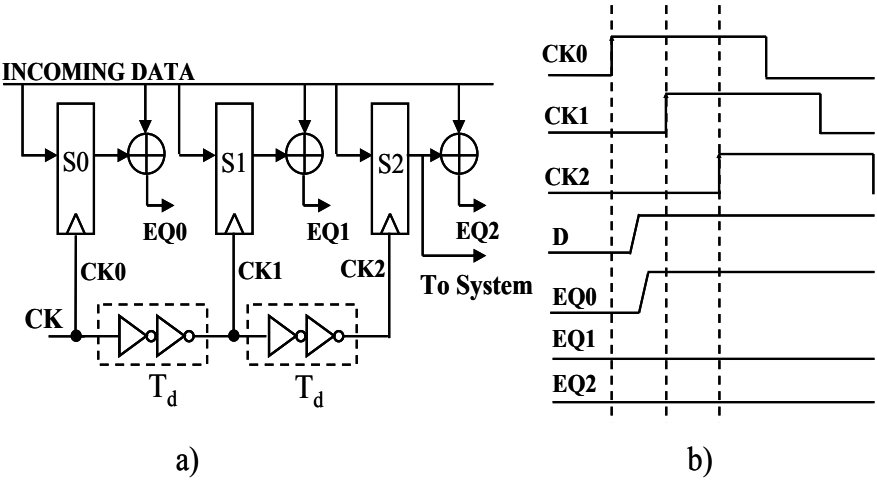


Figure 8.2 In situ monitoring: Kehl's triple-latch monitor. (a) In situ monitoring of delay. (b) Timing diagrams showing when system is "tuned".

to avoid overly aggressive clocking, evaluations of the latch values must be limited to tests using worst-case latency vectors. Kehl suggests that the system should periodically stop and test worst-case vectors to determine whether the system requires tuning. This requirement severely limits the general applicability of this approach since writing vectors that account for the worst-case delay and coupling noise scenario are difficult to generate and exercise for general-purpose processors.

8.2.4 Micro-architectural Techniques

A potential shortcoming of all the techniques discussed above is that they seek to track variations in the critical-path delay and, consequently, cannot adapt to input vector-dependent delay variations. The processor voltage and frequency are still limited by the worst-case path, even if it is not being sensitized. This issue is addressed by several micro-architectural techniques discussed in literature, specifically related to adder architectures [8] [20]. Such designs exploit the fact that the worst-case carry-chain length is rarely sensitized. This allows them to operate the adder block at a higher frequency than what is dictated by the worst-case carry path. If a latency-intensive add operation is detected, then the clock frequency is halved to allow it to complete without errors. An example of such a design is the stutter adder [8] which uses a low-overhead circuit for a priori determination of the carry-chain length. If the carry-chain length in a cycle exceeds a certain number of bits, then a "stutter" signal is raised which clock-gates

the next cycle. Thus a "long" adder computation is effectively given two cycles to execute. Recent studies [21] on SPECInt 2000 benchmarks have shown that the maximum carry-chain length for 64-bit additions rarely exceeds 24 bits. In fact, in [8], the authors report that in 95% of cases, the adder required only one cycle to compute. Lu [20] proposes a similar technique where an "approximate" but faster implementation of a functional unit is used in conjunction with a slow but always-correct checker to exploit typical latencies and clock the system a higher rate.

Data-dependent delay variations are also exploited by non-uniform cache architectures (NUCA) [22][23]. In aggressively scaled technologies, interconnect delay can become a significant portion of the cache access time. This causes wide variations in the fetch latencies of data words located near the access port versus those located further off. In traditional cache designs, the worst-case latency limits the cache access time. However, NUCA allows early access times for addresses near the access port, thereby achieving throughput improvement. Additional throughput can be achieved by mapping frequently accessed data to banks located nearest to the access port. Thus, in the context of NUCA, data dependence of delay relates to the frequency with which an address in the cache is accessed.

While the stutter adder and the NUCA architectures adapt to data-dependent variations, they still require margins to account for slow silicon grade and worst-case ambient conditions. On the contrary, error detection and correction approaches seek to achieve both, i.e., eliminate worst-case safety margins for all types of uncertainties and adapt to data-dependent variations as well. However, they are more complex and incur additional overhead in their implementation. Such approaches are discussed in detail in the next section.

8.3 Error Detection and Correction Approaches

The key concept of these schemes is to scale the system parameters (e.g., voltage and frequency) until the point where the processor fails to meet timing. A detection block flags the occurrence of a timing error after which a correction block is engaged to recover the correct state. To ensure that the system does not face persistent errors, an additional controller monitors the error rate and tunes voltage and frequency to achieve a targeted error rate.

Allowing the processor to fail and then recover helps eliminate worst-case safety margins. This enables significantly greater performance and energy efficiency over "always-correct" techniques. Furthermore, by tuning

system parameters based on the error rate, it is possible to exploit the input vector dependence of delay as well. Instead of safety margins, such systems rely on successful detection and correction of timing errors to guarantee computational correctness. The net energy consumption of the system is essentially a trade-off between the increased efficiency afforded by the elimination of margins and the additional overhead of recovery. Of course, the overhead of recovery can make sustaining a high error rate counterproductive. Hence, these systems typically rely on restricting operation to low error rate regimes to maximize energy efficiency.

Their relative complexity makes the general applicability of such systems difficult. However, they are naturally amenable for certain applications areas such as communications and signal processing. Communication systems require error correction to reliably transfer information across a noisy channel. Therefore, it is relatively easier to overload the existing error correction infrastructure to enable adaptivity to variable silicon and ambient conditions. Self-calibrating interconnects by Worm et al. [24] and algorithmic noise tolerance by Shanbhag et al. [25] are examples of applications of such techniques to on-chip communication and signal processing architectures.

8.3.1 Techniques for Communication and Signal Processing

Self-calibrating interconnects address the problem of reliable on-chip communication in aggressively scaled technologies. Signal integrity concerns require on-chip busses to be strongly buffered which consumes a significant portion of the total chip power. Hence, it is desirable to transfer bits at the lowest possible operating voltage while still guaranteeing the required performance and the targeted bit error rate (BER). Worm [24] addresses this issue by encoding the data words with the so-called self-synchronizing codes [24] before transmission. The receiver is augmented with a checker unit that decodes the received code word and flags timing errors. Correction occurs by requesting re-transmission through an automatic repeat request (ARQ) block, as shown in Figure 8.3. Furthermore, an additional controller obtains feedback from the checker and accordingly adjusts the voltage and the frequency of the transmission. By reacting to the error rates, the controller is able to adapt to the operating conditions and thus eliminate worst-case safety margins. This improves the energy efficiency of the on-chip busses with negligible BER degradation.

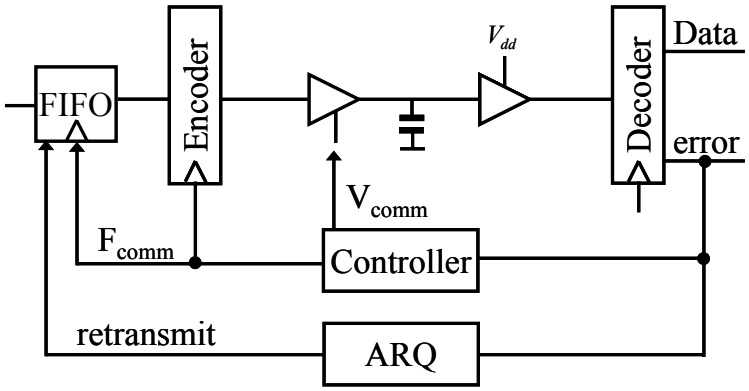


Figure 8.3 Self-calibrating interconnects.

Algorithmic noise tolerance (ANT) by Shanbhag et al. [25] uses a similar concept for low-power VLSI signal processing architectures. As conceptually illustrated in Figure 8.4, the main processor block is augmented with an estimator block. The main block is voltage scaled beyond the point of failure, thereby leading to intermittent timing errors. The result of the main block is validated against the result of the estimator block which computes correct result, based on the previous history. The estimator block is significantly cheaper in terms of area and power as compared to the main block which is being voltage scaled. At low error rates, the energy savings of aggressive scaling on the main block compensates for the overhead of correction, leading to significant energy savings. Error detection occurs when the difference in results of the main block and the estimator block exceeds a certain threshold. Error correction occurs by overwriting the result of the main block with that of the estimator block.

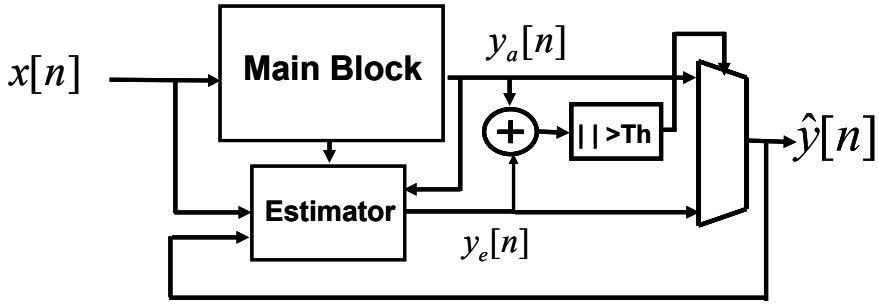


Figure 8.4 Algorithmic noise tolerance [25].

Since the estimator block depends on past history of correct results to make its prediction, its accuracy reduces as more errors are experienced. This adversely affects the BER of the entire block. In addition, the overhead of error correction also increases with increased error rate. Hence, it is desirable to keep the rate of timing errors low for maintaining a low BER and high energy efficiency. The authors built a FIR block in 0.35 micron technology [25] to demonstrate the efficacy of this technique. They obtained at least 70% savings over an error-free design for a 1% reduction in the signal-to-noise (SNR) ratio of the final output.

By reacting to error rates, both of the above techniques are able to exploit data-dependent delay variations because even under aggressively scaled voltage and frequency conditions, it is possible to maintain a low error rate as long as the critical paths are not being sensitized.

8.3.2 Techniques for General-Purpose Computing

Communication applications are inherently suited for error detection and correction techniques. Unfortunately, the same cannot be said for general-purpose computing. The key requirement for general-purpose computing is that the committed architectural state is always correct. Therefore, all timing errors that can possibly propagate to the architectural state have to be flagged and corrected. This is not an issue in communication because it does not affect the correct functionality of the system and leads to a negligible degradation of the BER, at worst. Unlike in communication, a timing error in the architectural state in general-purpose computing is a catastrophic failure and needs to be avoided at all costs. It is for this reason that there have been only a few examples of error detection and correction techniques in the area of general-purpose computing. Typically, such techniques rely on temporal redundancy for error detection. It was shown by Roberts et al. [26] that multi-bit bidirectional bit-flips occur in multiplier outputs under aggressive voltage scaling. Application of error-correcting codes for processor circuits to detect and correct such failures is infeasible due to the prohibitive area overhead incurred [26].

Using temporal redundancy for timing error detection requires two different samples of the monitored signal. The earlier speculative sample is validated against the latter “always-correct” version which is sampled according to conventional worst-case assumptions. The idea of temporal redundancy for error detection has been used extensively in the design and test community for at-speed delay testing. Anghel and Nicolaidis [27] use a similar concept for detecting SEU failures in combinational logic. A cosmic particle strike in the combinational logic manifests itself as a pulse

which can get captured by downstream flip-flops. The authors detect such an event by re-sampling the flip-flop input after the pulse has died down. A discrepancy between the two samples indicates a SEU event in the combinational logic.

Razor [28] uses temporal redundancy for general-purpose computing. In Razor, a critical-path signal is speculatively sampled at the rising edge of the regular clock and is compared against a shadow latch which samples at a delayed edge. A timing error is flagged when the speculative sample does not agree with the delayed sampled. State correction involves overwriting the shadow latch data into the main flip-flop and engaging micro-architectural recovery features to recover correct state. As is common with most error detection and correction techniques, Razor is able to eliminate worst-case safety margins by allowing errors to occur and recovering from them. We discuss Razor in greater detail in the next section onward.

8.4 Introduction to Razor

Razor [28] is a circuit-level timing speculation technique based on dynamic detection and correction of speed path failures in digital designs. As mentioned in the previous section, a critical-path signal in Razor is sampled twice. The earlier sample is speculatively consumed by the pipeline downstream logic. A timing error is flagged by comparing the speculative sample against the correct, later sample. In such an event, suitable recovery mechanisms are engaged to achieve correct state. In situ detection and recovery ensures correct operation and allows for the elimination of worst-case safety margins. Thus, with Razor, it is possible to tune the supply voltage to the level where first delay errors happen. In addition, voltage can also be scaled below this first point of failure into the sub-critical regime, thereby deliberately tolerating a targeted error rate. Due to the strong data dependence of circuit delay, only a few critical instructions are expected to fail while a majority will operate correctly. Razor automatically exploits this by tuning the supply voltage to obtain a small, but non-zero error rate. Of course, error correction adds energy overhead but this is minimal at low error rates. The extra voltage scaling headroom enabled by sub-critical operation enables substantial energy savings.

The fundamental trade-off that exists between power overhead of error correction against additional power savings from operating at a lower supply voltage is qualitatively illustrated in Figure 8.5. The point of first failure of the processor (V_{ff}) and the minimum allowable voltage of traditional DVS techniques (V_{margin}) are also labeled in the figure. V_{margin} is much

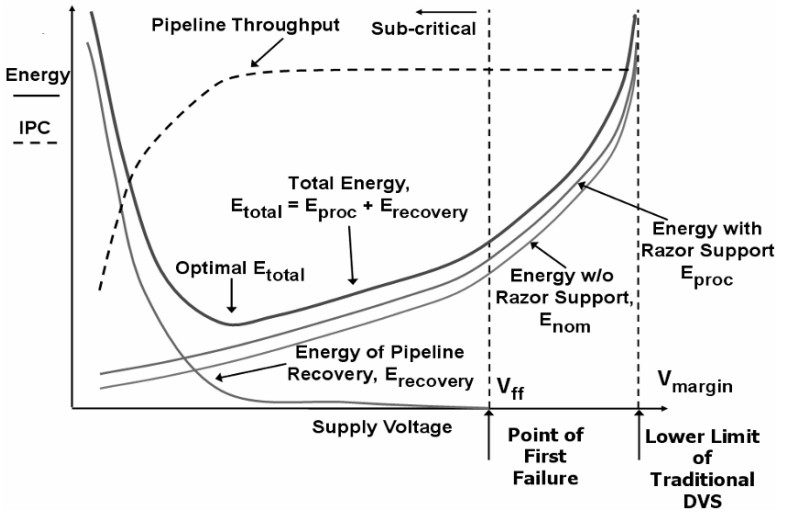


Figure 8.5 Qualitative relationship of energy and IPC as a function of supply voltage. (© IEEE 2005)

higher than V_{ff} under typical conditions, since safety margins need to be included to accommodate for worst-case operating conditions. In situ error detection and correction capability enables Razor to operate at V_{ff}, rather than at V_{margin}. The total energy of the processor (E_{tot}) is the sum of the energy required to perform standard processor operations (E_{proc}) and the energy consumed in recovery from timing errors ($E_{recovery}$). Of course, implementing Razor incurs power overhead due to which the nominal processor energy (E_{nom}) without Razor technology is slightly less than E_{proc}.

As the supply voltage is scaled, the processor energy (E_{proc}) reduces quadratically with voltage. However, as voltage is scaled below the first failure point (V_{ff}), a significant number of paths fail to meet timing. Hence, the error rate and the recovery energy ($E_{recovery}$) increase exponentially. The processor throughput also reduces due to the increasing error rate because the processor now requires more cycles to complete the instructions. The total processor energy (E_{tot}) shows an optimal point where the rate of change of $E_{recovery}$ and E_{proc} offsets each other.

8.4.1 Razor Error Detection and Recovery Scheme

Error detection in Razor occurs by augmenting the standard, positive edge-triggered critical-path flip-flops with a so-called shadow latch that samples

off the negative edge of the clock. Figure 8.6 shows the conceptual representation of a Razor flip-flop, henceforth referred to as the RFF. The input data is given additional time, equal to the duration of the positive clock phase, to settle down to its correct state before being sampled by the shadow latch. To ensure that the shadow latch always captures the correct data, the minimum allowable supply voltage needs to be constrained during design time such that the setup time at the shadow latch is never violated, even under worst-case conditions. A comparator flags a timing error when it detects a discrepancy between the speculative data sampled at the main flip-flop and the correct data sampled at the shadow latch.

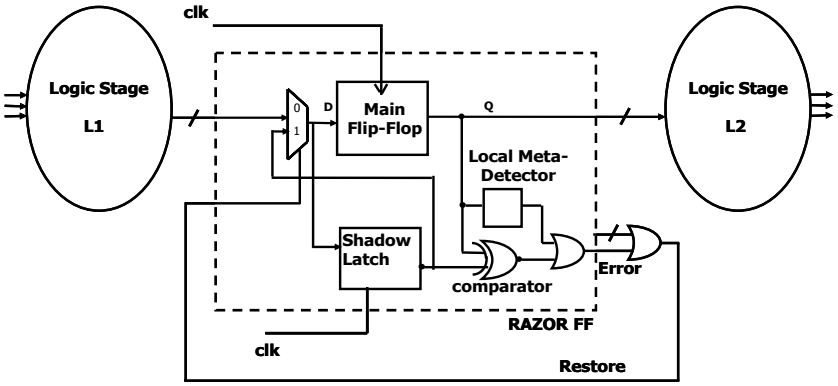


Figure 8.6 Razor flip-flop conceptual schematic. (© IEEE 2005)

Since setup and hold constraints at the main flip-flop input (*D*) are not respected, it is possible that the state of the flip-flop becomes metastable. A metastable signal increases critical-path delay which can cause a shadow latch in the succeeding pipeline stage to capture erroneous data, thereby leading to incorrect execution. In addition, a metastable flip-flop output can be inconsistently interpreted by the error comparator and the downstream logic. Hence, an additional detector is required to correctly flag the occurrence of metastability at the output of the main flip-flop. The outputs of the metastability detector and the error comparator are OR-ed to generate the *error* signal of the RFF. Thus, the system reacts to the occurrence of metastability in exactly the same way as a conventional timing failure.

A key point to note is the fact that metastability *need not be resolved correctly* in the RFF and that just the *detection* of such an occurrence is sufficient to engage the Razor recovery mechanism. However, in order to prevent potentially metastable signals from being committed to memory, at least two successive non-critical pipeline stages are required immediately before storage. This ensures that every signal is validated by Razor and is effectively double-latched in order to have a negligible probability of

being metastable, before being written to memory. In our design, data accesses in the memory stage were non-critical and hence we required only one additional pipeline stage to act as a dummy stabilization stage.

Error signals of individual RFFs are OR-ed together to generate the pipeline *restore* signal which overwrites the shadow latch data into the main flip-flop, thereby restoring correct state in the cycle following the erroneous cycle. Thus, an erroneous instruction is guaranteed to recover with a single cycle penalty, without having to be re-executed. This ensures that forward progress in the pipeline is always maintained. Even if every instruction fails to meet timing, the pipeline still completes, albeit at a slower speed. Upon detection of a timing error, a micro-architectural recovery technique is engaged to restore the whole pipeline to its correct state.

8.4.2 Micro-architectural Recovery

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 counter-flow pipelining [29].

8.4.2.1 Recovery Using 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, as shown in Figure 8.7(a). The additional clock period allows every stage to recompute its result using the Razor shadow latch as input. Consequently, any previously forwarded erroneous values will be replaced with the correct value from the Razor shadow latch, thereby guaranteeing forward progress. If all stages produce an error each cycle, the pipeline will continue to run, but at half the normal speed. To ensure negligible probability of failure due to metastability, there must be two non-speculative stages between the last Razor latch and the writeback (WB) stage. Since memory accesses to the data cache are non-speculative in our design, only one additional stage labeled ST (stabilize) is required before writeback (WB). In the general case, processors are likely to have critical memory accesses, especially on the read path. Hence, the memory sub-system needs to be suitably designed such that it can handle potentially critical read operations.

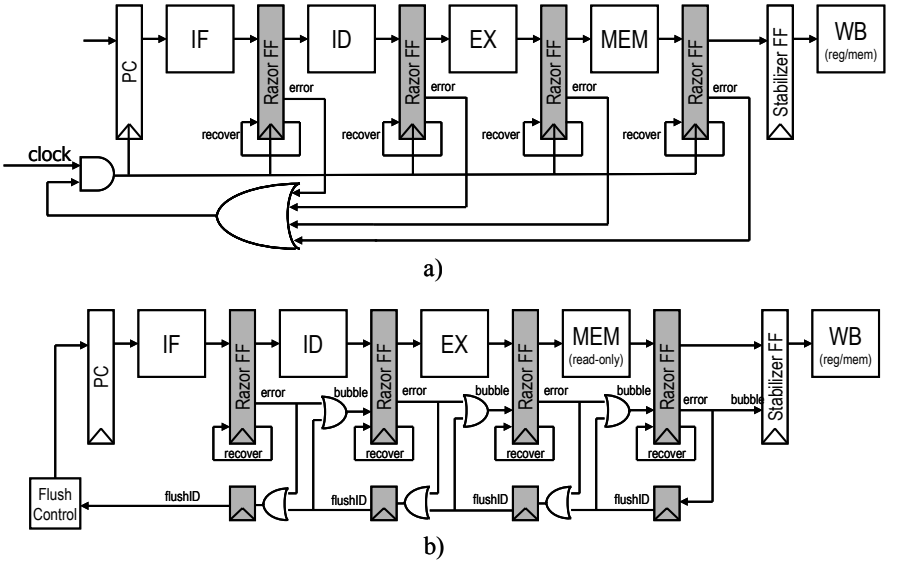


Figure 8.7 Micro-architectural recovery schemes. (a) Centralized scheme based on clock-gating. (b) Distributed scheme based on pipeline flush. (© IEEE 2005)

8.4.2.2 Recovery Using Counter-Flow Pipelining

In aggressively clocked designs, it may not be possible to implement single cycle, global clock-gating without significantly impacting processor cycle time. Consequently, we have designed and implemented a fully pipelined error recovery mechanism based on counter-flow pipelining techniques [29]. The approach illustrated in Figure 8.7(b) 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 erroneous 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 erroneous 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. When the flush ID reaches the start of the pipeline, the flush control logic restarts the pipeline at the instruction following the erroneous instruction.

8.4.3 Short-Path Constraints

The duration of the positive clock phase, when the shadow latch is transparent, determines the sampling delay of the shadow latch. This constrains the minimum propagation delay for a combinational logic path terminating in a RFF to be at least greater than the duration of the positive clock phase and the hold time of the shadow latch. Figure 8.8 conceptually illustrates this minimum delay constraint. When the RFF input violates this constraint and changes state before the negative edge of the clock, it corrupts the state of the shadow latch. Delay buffers are required to be inserted in those paths which fail to meet this minimum path delay constraint imposed by the shadow latch.

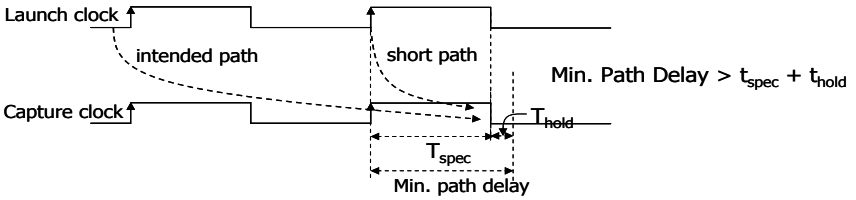


Figure 8.8 Short-path constraints.

The shadow latch sampling delay represents the trade-off between the power overhead of delay buffers and the voltage margin available for Razor sub-critical mode of operation. A larger value of the sampling delay allows greater voltage scaling headroom at the expense of more delay buffers and vice versa. However, since Razor protection is only required on the critical paths, overhead due to Razor is not significant. On the Razor prototype subsequently presented, the power overhead due to Razor was less than 3% of the nominal power overhead.

8.4.4 Circuit-Level Implementation Issues

Figure 8.9 shows the transistor level schematic of the RFF. The error comparator is a semi-dynamic XOR gate which evaluates when the data latched by the slave differs from that of the shadow in the negative clock phase. The error comparator shares its dynamic node, *Err_dyn,* with the metastability detector which evaluates in the positive phase of the clock when the slave output could become metastable. Thus, the RFF *error* signal is flagged when either the metastability detector or the error comparator evaluates.

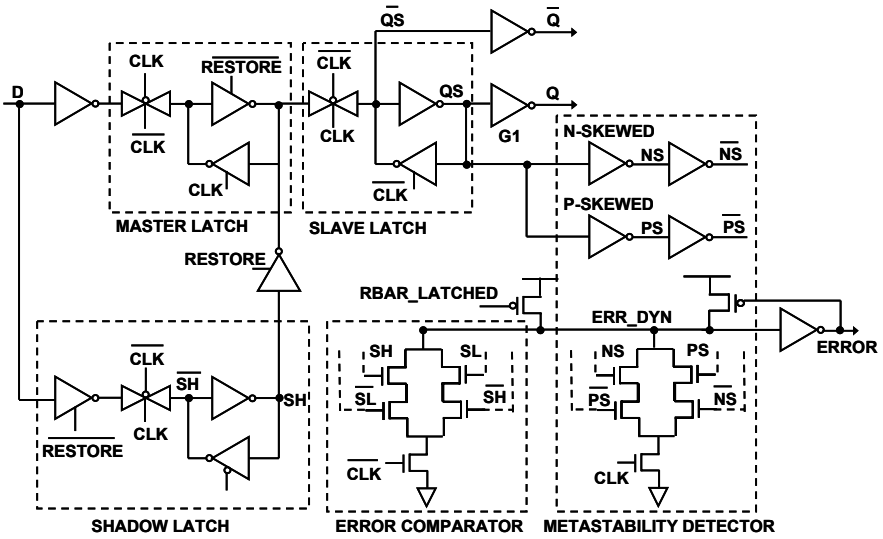


Figure 8.9 Razor flip-flop circuit schematic. (© IEEE 2005)

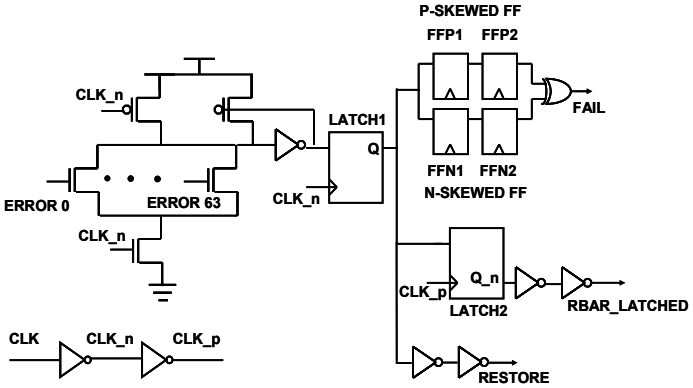


Figure 8.10 Restore generation circuitry. (© IEEE 2005)

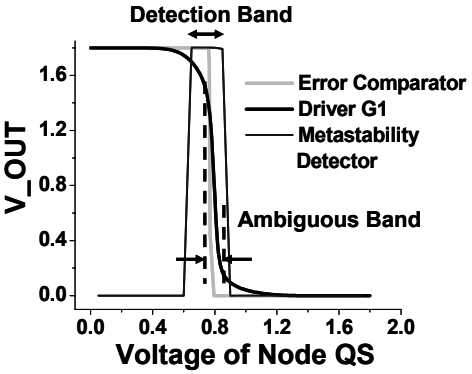
This, in turn, evaluates the dynamic gate to generate the *restore* signal by "OR"-ing together the error signals of individual RFFs (Figure 8.10), in the negative clock phase. The *restore* needs to be latched at the output of the dynamic OR gate so that it retains state during the next positive phase (recovery cycle) during which it disables the shadow latch to protect state. The shadow latch can be designed using weaker devices since it is required only for runtime validation of the main flip-flop data and does not form a part of the critical path of the RFF.

The *rbar_latched* signal, shown in the restore generation circuitry in Figure 8.10, which is the half-cycle delayed and complemented version of

the *restore* signal, precharges the *Err_dyn* node for the next errant cycle. Thus, unlike standard dynamic gates where precharge takes place every cycle, the *Err_dyn* node is conditionally precharged in the recovery cycle following a Razor error.

Compared to a regular DFF of the same drive strength and delay, the RFF consumes 22% extra (60fJ/49fJ) energy when sampled data is static and 65% extra (205fJ/124fJ) energy when data switches. However, in the processor, only 207 flip-flops out of 2388 flip-flops, or 9%, could become critical and needed to be RFFs. The Razor power overhead was computed to be 3% of nominal chip power.

The metastability detector consists of p- and n-skewed inverters which switch to opposite power rails under a metastable input voltage. The detector evaluates when input node SL can be ambiguously interpreted by its fan-out, inverter G1 and the error comparator. The DC transfer curve (Figure 8.11a) of inverter G1, the error comparator and the metastability detector show that the "detection" band is contained well within the ambiguously interpreted voltage band. Figure 8.11(b) gives the error detection and ambiguous interpretation bands for different corners. The probability that metastability propagates through the error detection logic and causes metastability of the restore signal itself was computed to be below 2e-30 [30]. Such an event is flagged by the fail signal generated using double-skewed flip-flops. In the rare event of a fail, the pipeline is flushed and the supply voltage is immediately increased.



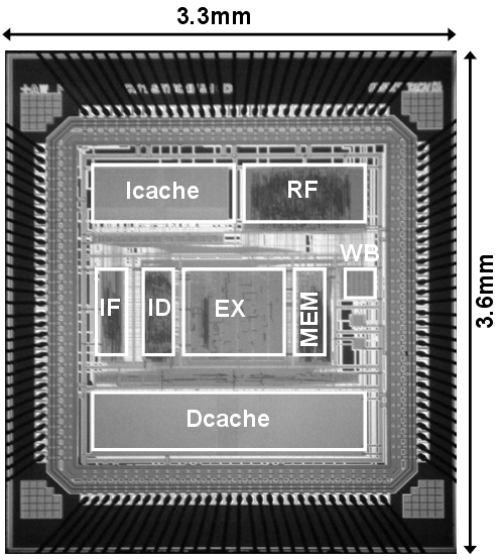
Corner			Ambiguous Band	Detection Band
Proc	VDD	TEMP		
Slow	1.2V	85C	0.57-0.60	0.53-0.64
Typ.	1.2V	40C	0.52-0.58	0.48-0.61
Fast	1.2V	27C	0.48-0.56	0.40-0.61
Slow	1.8V	85C	0.77-0.87	0.67-0.93
Typ.	1.8V	40C	0.71-0.83	0.65-0.90
Fast	1.8V	27C	0.64-0.81	0.58-0.89

b)

Figure 8.11 Metastability detector characteristics. (a) Principle of operation. (b) Metastability detector: corner analysis. (© IEEE 2005)

8.5 Silicon Implementation and Evaluation of Razor

A 64b processor which implements a subset of the Alpha instruction set was designed and built as an evaluation vehicle for the concept of Razor. The chip was fabricated with MOSIS [31] in an industrial 0.18 micron technology. Voltage control is based on the observed error rate and power savings are achieved by (1) eliminating the safety margins under nominal operating and silicon conditions and (2) scaling voltage 120mV below the first failure point to achieve a 0.1% targeted error rate. It was tested and measured for savings due to Razor DVS for 33 different dies from two different lots and obtained an average energy savings of 50% over the worst-case operating conditions by operating at the 0.1% error rate voltage at 120MHz. The processor core is a five-stage in-order pipeline which implements a subset of the Alpha instruction set. The timing critical stages of the processor are the Instruction Decode (ID) and the Execute (EX) stages. The distributed pipeline recovery scheme as illustrated in Figure 8.7(b) was implemented. The die photograph of the processor is shown in Figure 8.12(a), and the relevant implementation details are provided in Figure 8.12(b).



Technology Node	0.18µm
Max. Clock Frequency	140MHz
DVS Supply Voltage Range	1.2-1.8V
Total Number of Transistors	1.58million
Die Size	3.3mm*3.6 mm
Measured Chip Power at 1.8V	130mW
Icache Size	8KB
Dcache Size	8KB
Total Number of Flip-Flops	2388
Total Number of Razor Flip-Flops	207
Number of Delay Buffers Added	2801
Error Free Operation (Simulation Results)	
Standard FF Energy (Static/Switching)	49fJ/124fJ
RFF Energy (Static/Switching)	60fJ/205fJ
Total Delay Buffer Power Overhead	3.7mW
% Total Chip Power Overhead	2.9%
Error Correction and Recovery Overhead	
Energy of a RFF per error event	260fJ

a) b)

Figure 8.12 Silicon evaluation of Razor. (a) Die micrograph. (b) Processor implementation details. (© IEEE 2005)

8.5.1 Measurement Results

Figure 8.13 shows the error rates and normalized energy savings versus supply voltage at 120 and 140MHz for one of the 33 chips tested, henceforth referred to as chip1. Energy at a particular voltage is normalized with respect to the energy at the point of first failure. For all plotted points, correct program execution with Razor was verified. The *Y*-axis on the left shows the percentage error rate and that on the right shows the normalized energy of the processor.

From the figure, we note that the error rate at the point of first failure is very low and is of the order of 1.0e-7. At this voltage, a few critical paths that are rarely sensitized fail to meet setup requirements and are flagged as timing errors. As voltage is scaled further into the sub-critical regime, the error rate increases exponentially. The IPC penalty due to the error recovery cycles is negligible for error rates below 0.1%. Under such low error rates, the recovery overhead energy is also negligible and the total processor energy shows a quadratic reduction with the supply voltage. At error rates exceeding 0.1%, the recovery energy rapidly starts to dominate, offsetting the quadratic savings due to voltage scaling. For the measured chips, the energy optimal error rate fell at approximately 0.1%.

The correlation between the first failure voltage and the 0.1% error rate voltage is shown in the scatter plot of Figure 8.14. The 0.1% error rate voltage shows a net variation of 0.24V from 1.38V to 1.62V which is approximately 20% less than the variation observed for the voltage at the point of

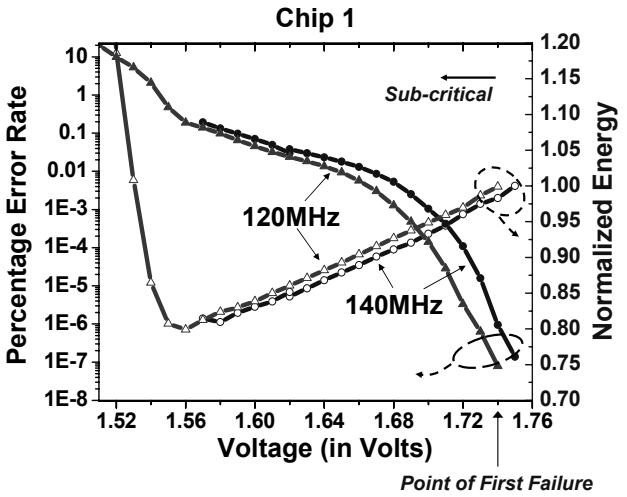


Figure 8.13 Measured error rate and energy versus supply voltage. (© IEEE 2005)

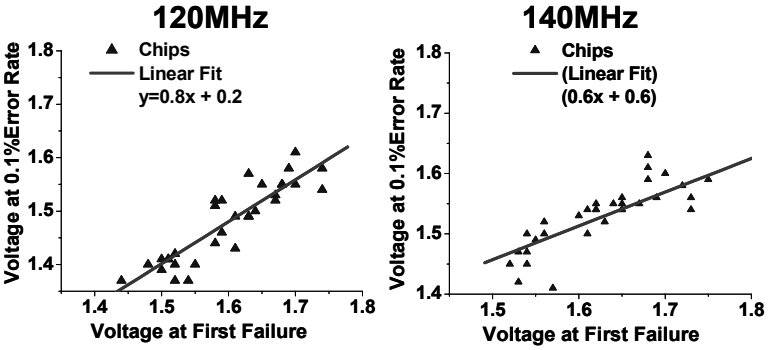


Figure 8.14 Scatter plot showing the point of 0.1% error rate versus the point of first failure. (© IEEE 2005)

first failure. The relative "flatness" of the linear fit indicates less sensitivity to process variation when running at a 0.1% error rate than at the point of first failure. This implies that a Razor-enabled processor, designed to operate at the energy optimal point, is likely to show greater predictability in terms of performance than a conventional worst-case optimized design. The energy optimal point requires a significant number of paths to fail and statistically averages out the variations in path delay due to process variation, as opposed to the first failure point which, being determined by the single longest critical path, shows higher process variation dependence.

8.5.2 Total Energy Savings with Razor

The total energy savings was measured by quantifying the savings due to elimination of safety margins and operation in the sub-critical voltage regime. Table 8.2 lists the measured voltage margins for process, voltage and temperature uncertainties for 2 out of the 33 chips tested, when operating at 120MHz. The chips are labeled as chip 1 and chip 2, respectively. The first failure voltage for chips 1 and 2 are 1.74V and 1.63V, respectively, and hence represent slow and typical process conditions, respectively.

Table 8.2 Measurement of voltage safety margins.

	Margins		
Chip (point of first failure)	**Process**	**Voltage**	**Temperature**
Slowest chip (1.76V)	0mV	180mV	100mV
Chip 1 (1.73V)	30mV	180mV	100mV
Chip 2 (1.63V)	130mV	180mV	100mV

The point of first failure of the slowest chip at 25°C is 1.76V. For this chip to operate correctly in the worst-case, voltage and temperature margins are added over and above the first failure voltage. The worst-case temperature margin was measured as the shift in the point of first failure of this chip when heated from 25°C to 10°5C. At 105°C, this chip fails at 1.86V, an increase of 100mV over the first failure voltage at 25°C. The worst-case voltage margin was estimated to be 10% of the nominal supply voltage of 1.8V (180mV). The margin for inter-die process variations was measured as the difference in the point of first failure voltage of the chip under test and the slowest chip. For example, chip 2 fails at 1.63V at 25°C when compared with the slowest chip which fails at 1.76V. This translates to 130mV process margin. Thus, with the incorporation of 100mV temperature margin and 180mV voltage margin over the first failure point of the slowest chip, the worst-case operating voltage for guaranteed correct operation was obtained to be 2.04V.

Figure 8.15 lists the energy savings obtained through Razor for chips 1 and 2. The first set of bars shows the energy when Razor is turned off and the chip under test is operated at the worst-case operating voltage at 120MHz, as determined for all the chips tested. At the worst-case voltage of 2.04V, chip 2 consumes 160.5mW of which 27.3mW is due to 180mV margin for supply voltage drop, 11.2mW is due to 100mV temperature margin and 17.3mW is due to 30mV process margin.

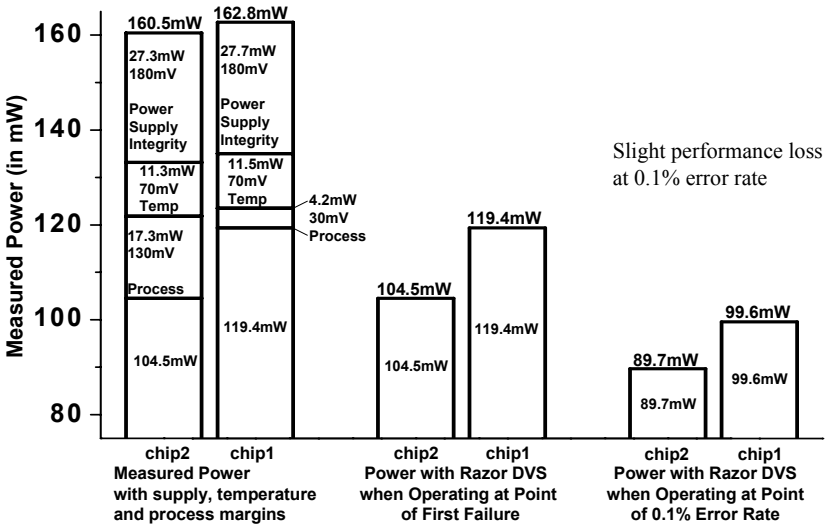


Figure 8.15 Total energy savings. (© IEEE 2005)

The second set of bars shows the energy when operating with Razor enabled at the point of first failure with all the safety margins eliminated. At the point of first failure, chip 2 consumes 104.5mW, while chip 1 consumes 119.4mW of power. Thus, for chip 2, operating at the first failure point leads to a saving of 56mW which translates to 35% saving over the worst case. The corresponding saving for chip 1 is 27% over the worst case.

The third set of bars shows the additional energy savings due to sub-critical mode of operation of Razor. With Razor enabled, both chips are operated at the 0.1% error rate voltage and power measurements are taken. At the 0.1% error rate, chip 1 consumes 99.6mW of power at 0.1% error rate which is a saving of 39% over the worst case. When averaged over all die, we obtain approximately 50% savings over the worst case at 120MHz and 45% savings at 140MHz when operating at the 0.1% error rate voltage.

8.5.3 Razor Voltage Control Response

Figure 8.16 shows the basic structure of the hardware control loop that was implemented for real-time Razor voltage control. A proportional integral algorithm was implemented for the controller in a Xilinx XC2V250 FPGA [32]. The error rate was monitored by sampling the on-chip error register at a conservative frequency of 750KHz. The controller reacts to the error rate that is monitored by sampling the error register and regulates the supply voltage through a DAC and a DC–DC switching regulator to achieve a targeted error rate. The difference between the sampled error rate and the targeted error rate is the error rate differential, E_{diff}. A positive value of E_{diff} implies that the CPU is experiencing too few errors and hence the supply voltage may be reduced and vice versa.

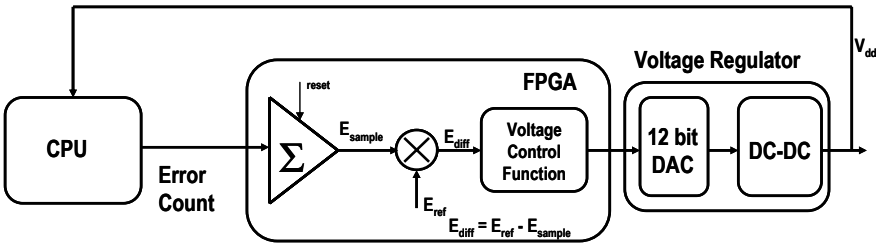


Figure 8.16 Razor voltage control loop. (© IEEE 2005)

The voltage controller response for a test program was tested with alternating high and low error rate phases. The targeted error rate for the given trace is set to 0.1% relative to CPU clock cycle count. The controller

response during a transition from the lowerror rate phase to the high-error rate phase is shown in Figure 8.17(a). Error rates increase to about 15% at the onset of the high-error phase. The error rate falls until the controller reaches a high enough voltage to meet the desired error rate in each millisecond sample period. During a transition from the high-error rate phase to the low-error rate phase, shown in Figure 8.17(b), the error rate drops to zero because the supply voltage is higher than required. The controller responds by gradually reducing the voltage until the target error rate is achieved.

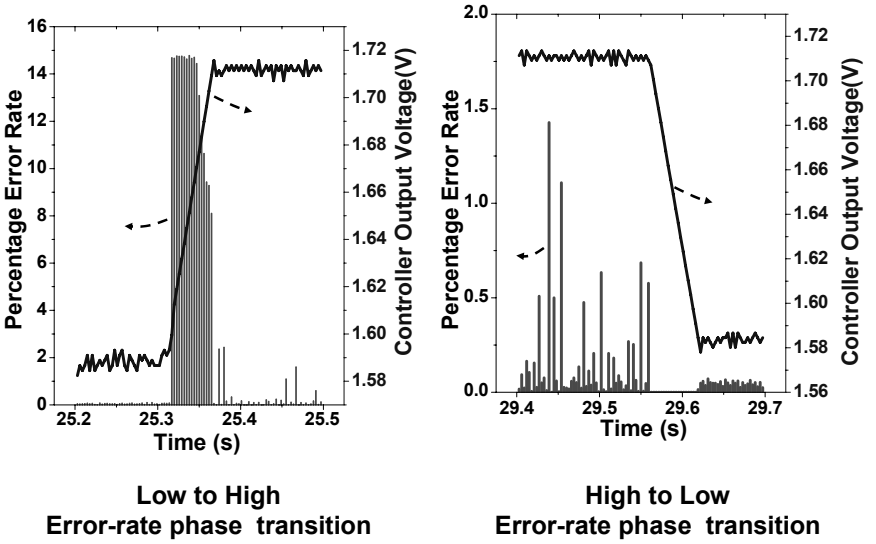


Figure 8.17 Voltage controller phase transition response. (a) Low to high transition. (b) High to low transition. (© IEEE 2005)

8.6 Ongoing Razor Research

Currently, research efforts on Razor are underway in ARM Ltd, UK. A deeper analysis of Razor as explained in the previous sections reveals several key issues that need to be addressed, before Razor can be deployed as mainstream technology.

The primary concern is the issue of Razor energy overhead. Since industrial strength designs are typically balanced, it is likely that significantly larger percentage of flip-flops will require Razor protection. Consequently, a greater number of delay buffers will be required to satisfy the short-path constraints. Increasing intra-die process variability, especially on the short paths, further aggravates this issue.

Another important concern is ensuring reliable state recovery in the presence of timing errors. The current scheme imposes a massive fan-out load on the pipeline *restore* signal. In addition, the current scheme cannot recover from timing errors in critical control signals which can cause undetectable state corruption in the shadow latch. Metastability on the *restore* signal further complicates state recovery. Though such an event is flagged by the *fail* signal, it makes validation and verification of a "Razor"-ized processor extremely problematic in current ASIC design methodologies.

An attempt is made to address these concerns by developing an alternative scheme for Razor, henceforth referred to as Razor II. The key idea in Razor II is to use the Razor flip-flop only for error detection. State recovery after a timing error occurs by a conventional replay mechanism from a check-pointed state. Figure 8.18 shows the pipeline modifications required to support such a recovery mechanism. The architectural state of the processor is check-pointed when an instruction has been validated by Razor and is ready to be committed to storage. The check-pointed state is buffered from the timing critical pipeline stages by several stages of stabilization which reduce the probability of metastability by effectively double-latching the pipeline output. Upon detection of a Razor error, the pipeline is flushed and system recovers by reverting back to the check-pointed architectural state and normal execution is resumed. Replaying from the

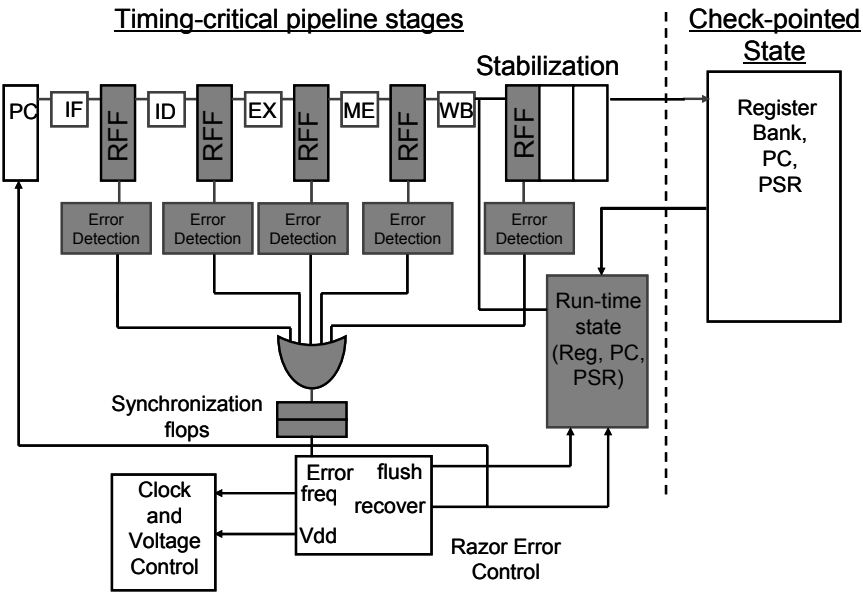


Figure 8.18 Pipeline modifications required for Razor II.

check-pointed state implies that a single instruction can fail in successive roll-back cycles, thereby leading to a deadlock. Forward progress in such a system is guaranteed by detecting a repeatedly failing instruction and executing the system at half the nominal frequency during recovery.

Error detection in Razor II is based on detecting spurious transitions in the D-input of the Razor flip-flop, as conceptually illustrated in Figure 8.19. The duration where the input to the RFF is monitored for errors is called the detection window. The detection window covers the entire positive phase of the clock cycle. In addition, it also includes the setup window in front of the positive edge of the clock. Thus, any transition in the setup window is suitably detected and flagged. In order to reliably flag potentially metastable events, safety margin is required to be added to the onset of the detection window. This ensures that the detection window covers the setup window under all process, voltage and temperature conditions. In a recent work, the authors have applied the above concept to detect and correct transient single event upset failures [33].

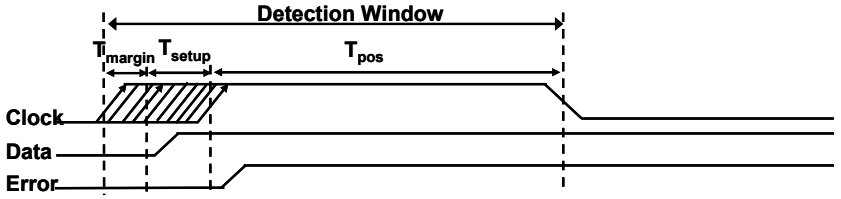


Figure 8.19 Transition detection-based error detection.

8.7 Conclusion

In this chapter, we presented a survey of different adaptive techniques reported in literature. We analyzed the concept of design margining in the presence of process variations and looked at how different adaptive techniques help eliminate some of the margins. We categorized these techniques as "always-correct" and "error detection and correction" techniques. We presented Razor as a special case study of the latter category and showed silicon measurement results on a chip using Razor for supply voltage control.

As process variations increase with each technology generation, adaptive techniques assume even greater relevance. However, deploying such techniques in the field is hindered either by their complexity as in the case

of Razor or by the lack of substantial gains as in the case of canary circuits. Future research in this field needs to focus on combining effectiveness of Razor in eliminating design margins with the relative simplicity of the "always-correct" techniques. As uncertainties worsen, adaptive techniques provide a solution toward achieving computational correctness and faster design closure.

References

[1] S.T. Ma, A. Keshavarzi, V. De, J.R. Brews, "A statistical model for extracting geometric sources of transistor performance variation," IEEE Transactions on Electron Devices, Volume 51, Issue 1, pp. 36–41, January 2004.

[2] R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and threshold voltage scaling for low power CMOS," IEEE Journal of Solid-State Circuits, Volume 32, Issue 8, August 1997.

[3] S. Yokogawa, H. Takizawa, "Electromigration induced incubation, drift and threshold in single-damascene copper interconnects," IEEE 2002 International Interconnect Technology Conference, 2002, pp. 127–129, 3–5 June 2002.

[4] W. Jie and E. Rosenbaum, "Gate oxide reliability under ESD-like pulse stress," IEEE Transactions on Electron Devices, Volume 51, Issue 7, July 2004.

[5] International Technology Roadmap for Semiconductors, 2005 edition, http://www.itrs.net/ Links/2005ITRS/Home2005.htm.

[6] M. Hashimoto, H. Onodera, "Increase in delay uncertainty by performance optimization," IEEE International Symposium on Circuits and Systems, 2001, Volume 5, pp. 379–382, 5, 6–9 May 2001.

[7] S. Rangan, N. Mielke and E. Yeh, "Universal recovery behavior of negative bias temperature instability," IEEE Intl. Electron Devices Mtg., p. 341, December 2003.

[8] G. Wolrich, E. McLellan, L. Harada, J. Montanaro, and R. Yodlowski, "A high performance floating point coprocessor," IEEE Journal of Solid-State Circuits, Volume 19, Issue 5, October 1984.

[9] Trasmeta Corporation, "LongRun Power Management," http://www.transmeta.com/tech/longrun2.html

[10] Intel Corporation, "Intel Speedstep Technology," http://www.intel.com/support/processors/mobile/pentiumiii/ss.htm

[11] ARM Limited, http://www.arm.com/products/esd/iem_home.html

[12] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," International Solid-State Circuits Conference, February 2000.

[13] A.K. Uht, "Going beyond worst-case specs with TEATime," IEEE Micro Top Picks, pp. 51–56, 2004

[14] K.J. Nowka, G.D. Carpenter, E.W. MacDonald, H.C. Ngo, B.C Brock, K.I. Ishii, T.Y. Nguyen and J.L. Burns, "A 32-bit powerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling," IEEE Journal of Solid-State Circuits, Volume 37, Issue 11, pp. 1441–1447, November 2002

[15] T.D. Burd, T.A. Pering, A.J. Stratakos and R.W. Brodersen, "A dynamic voltage scaled microprocessor system," IEEE Journal of Solid-State Circuits, Volume 35, Issue 11, pp. 1571–1580, November 2000

[16] Berkeley Wireless Research Center, http://bwrc.eecs.berkeley.edu/

[17] M. Nakai, S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano and M. Shimura, "Dynamic voltage and frequency management for a low power embedded microprocessor," IEEE Journal of Solid-State Circuits, Volume 40, Issue 1, pp. 28–35, January. 2005

[18] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Ngyugen, N. James and M. Floyd, "A distributed critical-path timing monitor for a 65nm high-performance microprocessor," International Solid-State Circuits Conference, pp. 398–399, 2007.

[19] T. Kehl, "Hardware self-tuning and circuit performance monitoring," 1993 Int'l Conference on Computer Design (ICCD-93), October 1993.

[20] S. Lu, "Speeding up processing with approximation circuits," IEEE Micro Top Picks, pp. 67–73, 2004

[21] T. Austin, V. Bertacco, D. Blaauw and T. Mudge, "Opportunities and challenges in better than worst-case design," Proceedings of the ASP-DAC 2005, Volume 1, pp. 18–21, 2005.

[22] C. Kim, D. Burger and S.W. Keckler, IEEE Micro, Volume 23, Issue 6, pp. 99–107, November–December 2003.

[23] Z. Chishti, M.D. Powell, T. N. Vijaykumar, "Distance associativity for high-performance energy-efficient non-uniform cache architectures," Proceedings of the International Symposium on Microarchitecture, 2003, MICRO-36

[24] F. Worm, P. Ienne and P. Thiran, "A robust self-calibrating transmission scheme for on-chip networks," IEEE Transactions on Very Large Scale Integration, Volume 13, Issue 1, January 2005.

[25] R. Hegde and N. R. Shanbhag, "A voltage overscaled low-power digital filter IC," IEEE Journal of Solid-State Circuits, Volume39, Issue 2, February 2004.

[26] D. Roberts, T. Austin, D. Blaauw, T. Mudge and K. Flautner, "Error analysis for the support of robust voltage scaling," International Symposium on Quality Electronic Design (ISQED), 2005.

[27] L. Anghel and M. Nicolaidis, "Cost reduction and evaluation of a temporary faults detecting technique," Proceedings of Design, Automation and Test in Europe Conference and Exhibition 2000, 27–30 March 2000 pp. 591–598

[28] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, T. Mudge, K. Flautner, "A self-tuning DVS processor using delay-error detection and correction," IEEE Journal of Solid-State Circuits, pp. 792–804, April 2006.

[29] R. Sproull, I. Sutherland, and C. Molnar, "Counterflow pipeline processor architecture," Sun Microsystems Laboratories Inc. Technical Report SMLI-TR-94-25, April 1994.
[30] W. Dally, J. Poulton, Digital System Engineering, Cambridge University Press, 1998
[31] www.mosis.org
[32] www.xilinx.com
[33] D. Blaauw, S.Kalaiselvam, K. Lai, W.Ma, S. Pant, C. Tokunaga, S. Das and D.Bull "RazorII: In-situ error detection and correction for PVT and SER tolerance," International Solid-State Circuits Conference, 2008
[34] D. Ernst, N. S. Kim, S. Das, S. Pant, T. Pham, R. Rao, C. Ziesler, D. Blaauw, T. Austin, T. Mudge, K. Flautner, "Razor: A low-power pipeline based on circuit-level timing speculation," Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 7–18, December 2003.
[35] A. Asenov, S. Kaya, A.R. Brown, "Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness," IEEE Transactions on Electron Devices, Volume 50, Issue 5, pp. 1254–1260, May 2003.
[36] K. Ogata, "Modern control engineering," 4th edition, Prentice Hall, New Jersey, 2002.