Razor and ReCycle

AMEEN AKEL

Razor

Razor – Motivation

- Power
  - Today's designs are extremely power hungry
    - Power now a limiting factor
  - Performance cannot be sacrificed (overall) to save on power
    - Both situations must continue improving
- Static Voltage Scaling
  - Not adaptive enough
  - Must be conservative estimates
    - Wastes power savings for little to no performance benefit
- Make average silicon matter!

Razor – Approach

- The Main Idea
  - Circuit delay is data dependent, so why should designers care about the conservative case?
    - Shooting for the "average" case – just like with ReCycle
  - Lower the supply voltage (to sub-critical voltages) to reduce power throughout the chip
  - What happens if execution encounters a "worst-case" path through a pipeline stage?
    - Wrong data can be latched and moved to the next stage
  - Razor hybrids a few previous designs to solve this.

Razor Design Goals

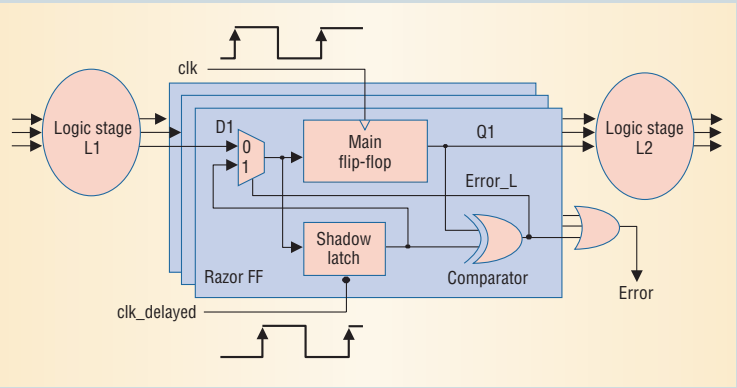
- Razor hardware must not interfere with error-free operation of a pipeline
  - Nearly invisible to the common case
- Razor hardware cannot fail; it must always be correct
- Razor hardware must be minimal in both hardware size and power footprint

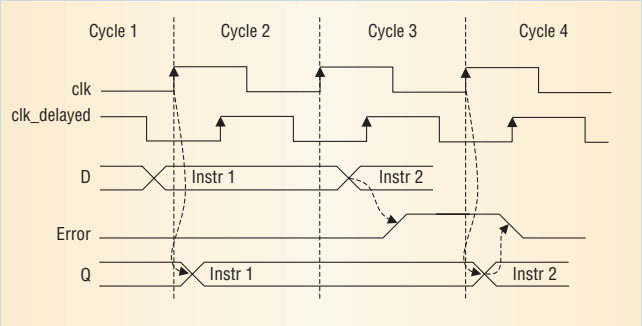
Razor – Approach

- What's unique to Razor?
  - Counterflow pipeline in the synchronous world
  - Handling metastability
  - Inducing error-prone state
- What did it inherit?
  - Delayed latch *idea* (Triple Latch), but not *implementation* (Shadow Latch)
  - Error correction method (from DIVA)

Razor – Approach

- Exploring the Shadow Latch
  - Method to detect and recover from errors in a minimal number of cycles
  - Shadow latch is delayed about 50% behind the main flip-flop's clock in order to catch any timing errors
  - A comparator will quickly decide how accurate the data in the flip-flop is via an XOR gate.
  - Pipeline stages are designed so that in the absolute worst case, the shadow latch's setup time is met.
  - An encountered error will invalidate any data coming out of the flip-flop for that cycle





Razor – Approach

- Metastability – The state in which a signal is neither 0 nor 1. The state usually settles around $V_{dd}/2$.
  - Shadow latch can never be metastable, based upon its timing constraints.
  - If flip-flop becomes metastable, the metastability detector can report on that fact (most of the time).
  - Small chance that Error can become metastable, which is claimed as inevitable. In this case, a panic signal is raised and the pipeline is flushed.

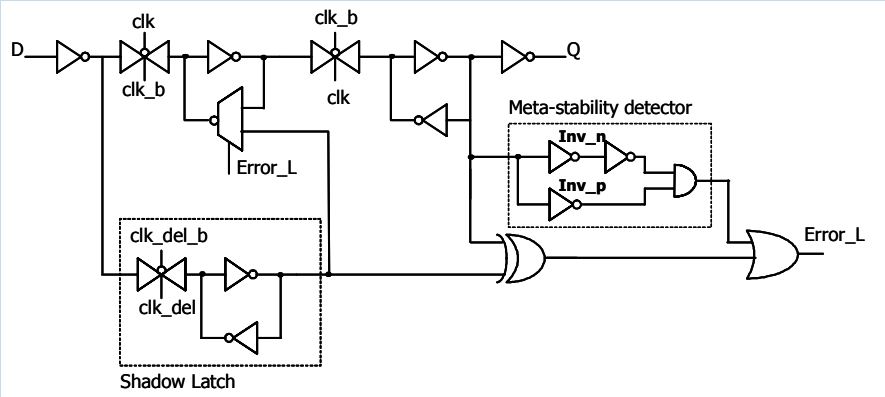
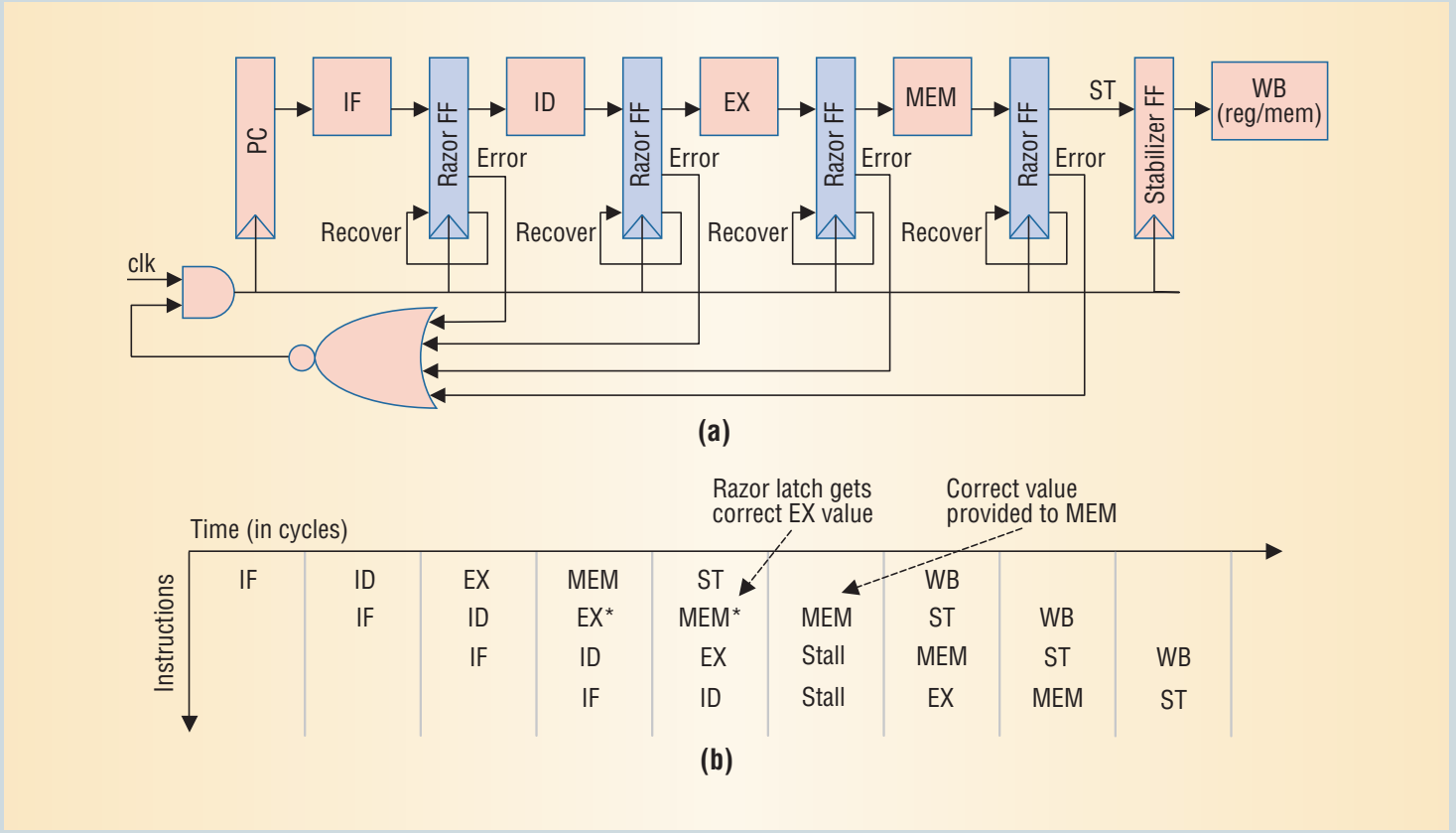


Figure 2. Reduced overhead Razor flip-flop and meta-stability detection circuits.

Razor Approach – Recovery

- Clock Gating

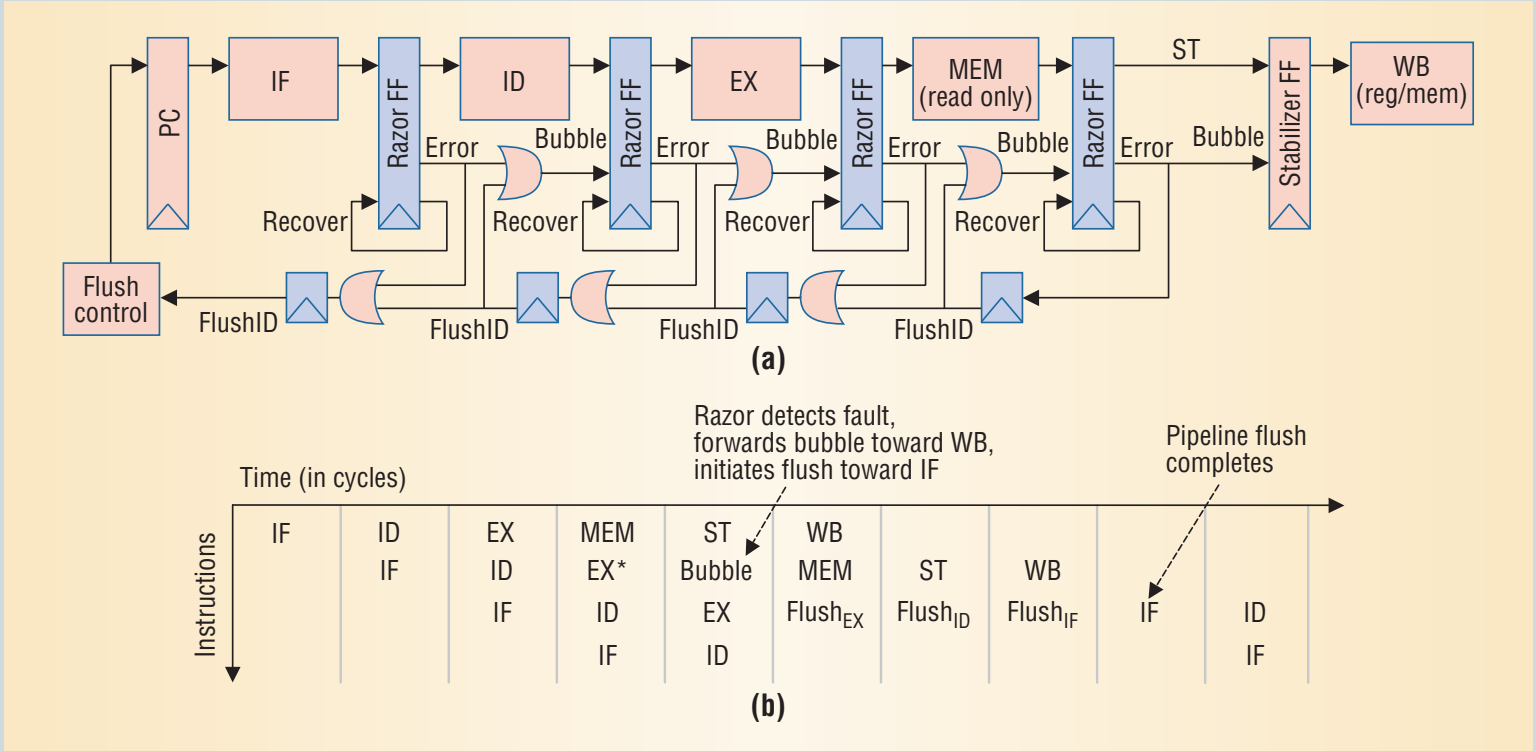


Razor Approach – Recovery

- Clock Gating
  - Pipeline stalls on any Razor error
  - Forward progress is guaranteed, as the problematic input is always available at the previous stage's Shadow Latch
    - Only a single cycle stall is required to recompute the next stage's value, and the pipeline can continue.
  - Possible long cycle time
    - Cycle time must be long enough so that any stage in the pipeline can deliver a clock gating signal to the rest of the Flip Flops.

Razor Approach – Recovery

- Counterflow



Razor Approach – Recovery

- Counterflow Pipelining
  - Uses an asynchronous-like design to propagate errors backwards
  - Now the error propagation is also pipelined, which translates to a minimal effect on the cycle time of each stage.
    - This translates into a tradeoff between resuming within one cycle versus a faster cycle time
  - Error signal travels through each pipelined register until reaching the PC, which then restarts execution.

Razor Approach – Dynamic Adjustments

- Focus on a constant error rate (E_{ref})
  - Change voltages based upon this measurement
- Pros
  - Real dynamic changes based on the runtime conditions
- Cons
  - Voltage regulators are slow
  - Slow reaction causes overcompensation

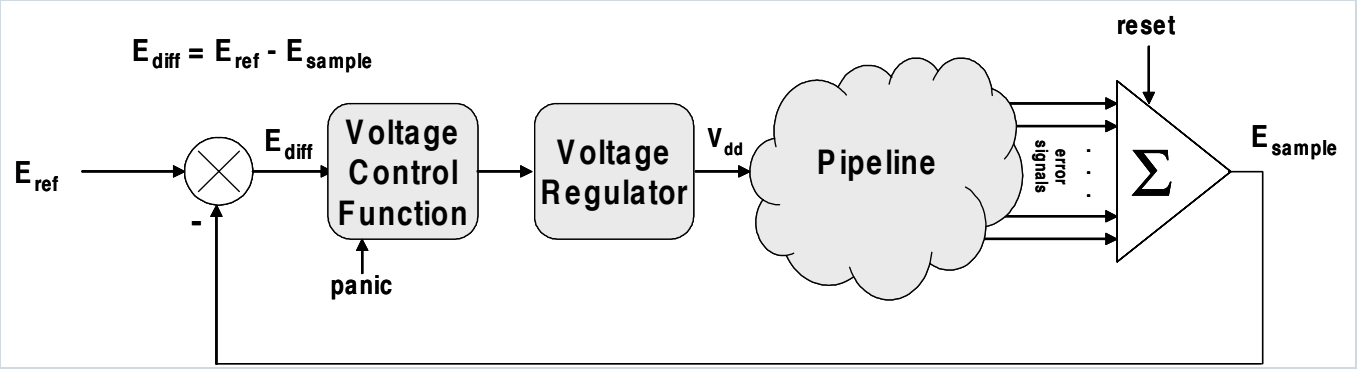


Figure 6. Supply Voltage Control System

Razor – Simulations/Data

- Alpha-64 Simulation
  - Parameters:
    - In-order pipeline
    - 8 KB I/D Caches
    - 192/2408 flip-flops were augmented with a shadow latch.
  - Important results:
    - 3.1% total power overhead for Razor parts
    - 1% of total power for recovery overhead

Razor – Simulations/Data

- FPGA Multiplier Simulation

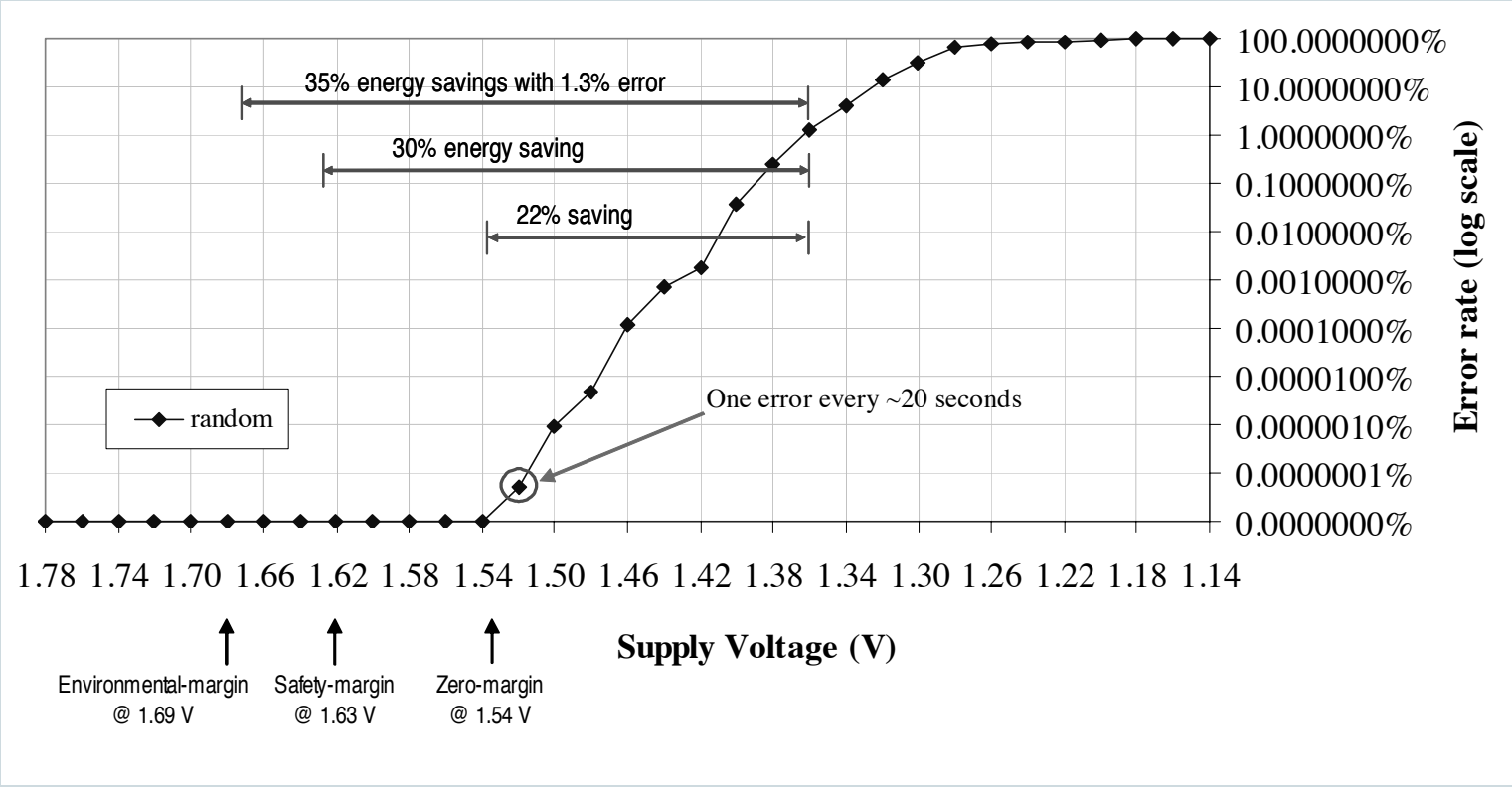


Figure 9. Measured Error Rates for an 18x18-bit FPGA Multiplier Block at 90 MHz and 27 C.

Razor – Simulations/Data

- Adder Simulation
  - Fixed voltage sweep
  - Goal:
    - Reduce energy without sacrificing IPC

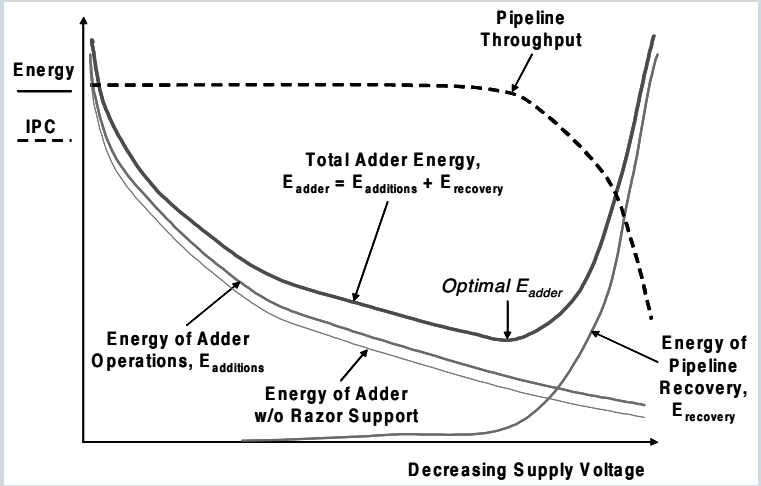


Figure 11. The Qualitative Relationship Between Supply Voltage, Energy and Pipeline Throughput (for a fixed frequency).

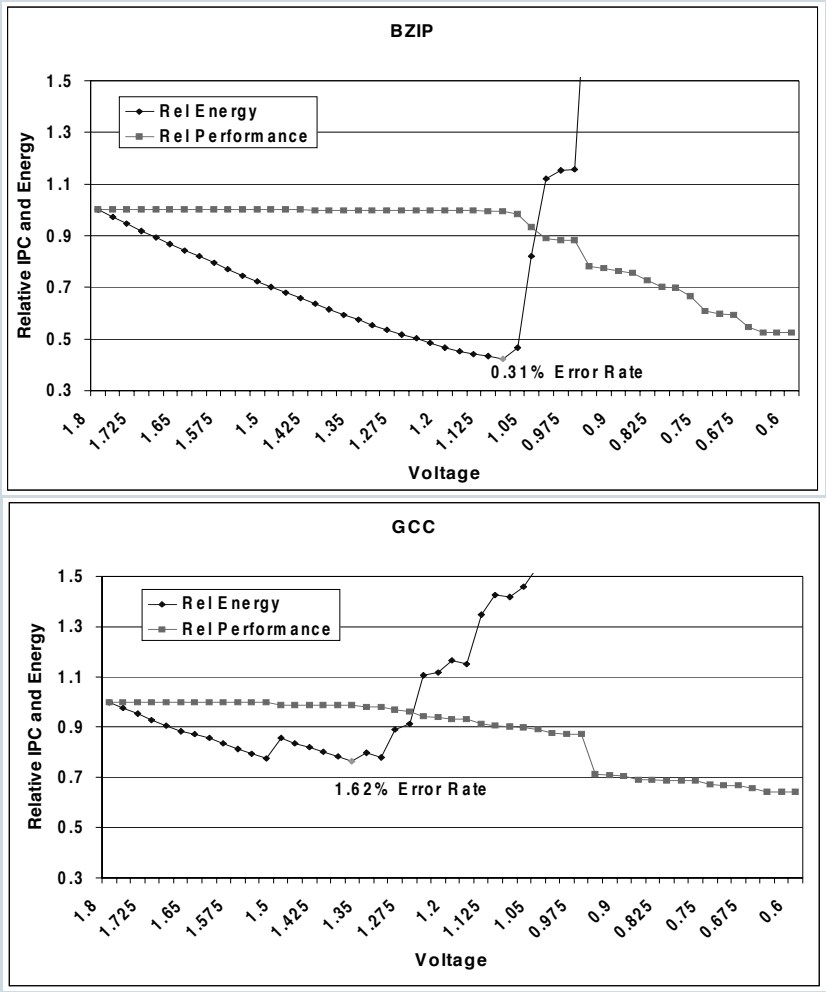


Figure 12. Relative Adder Energy and Pipeline Throughput for Simulated Benchmarks.

Razor – Simulations/Data

- Dynamic Scaling
  - Target error rate was 1.5%
  - Takes 5000 cycle chunk samples
  - Uses those chunks to dynamically scale voltage
- Slow reaction times

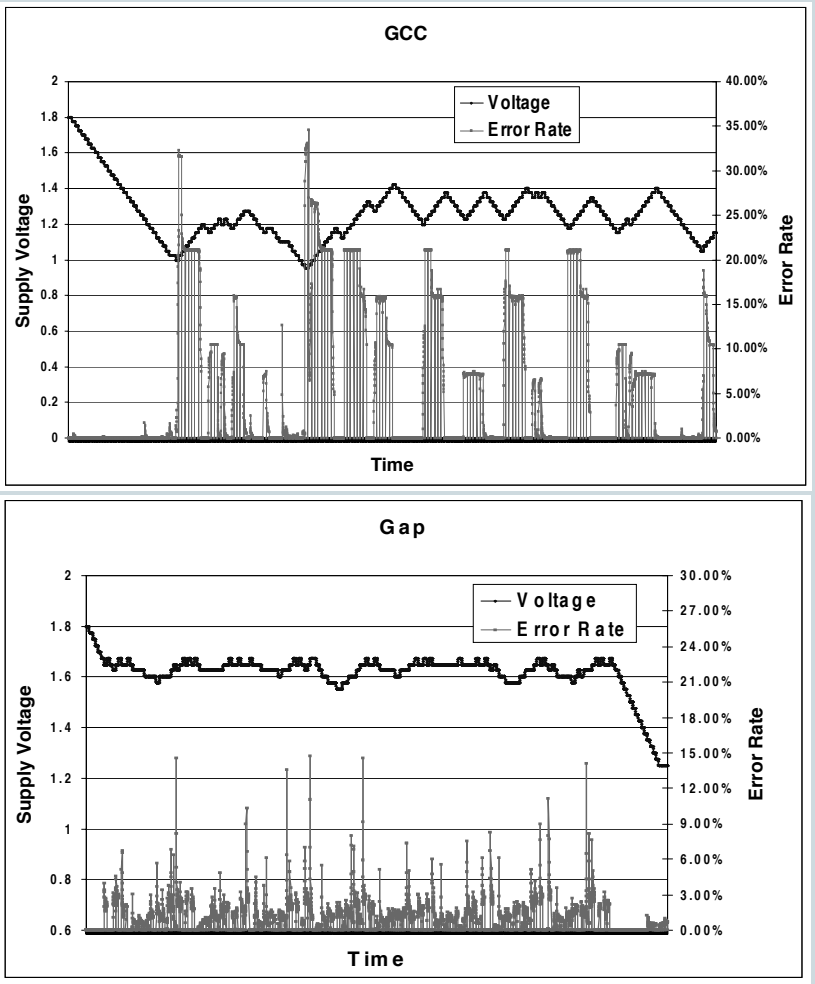


Figure 13. Adder Error Rate and Voltage Controller Response.

Razor – Simulations/Data

- Mixed results
  - Half see increases, half see decreases in energy relative to static scaling

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%*	

Razor – Conclusions

- Similarities to DIVA
  - Error checking component becomes an oracle
  - Added error checking does not interfere with original pipeline other than the case of an error
- Differences from DIVA
  - Does not handle transient errors; only handles timing errors

ReCycle

ReCycle – Motivation

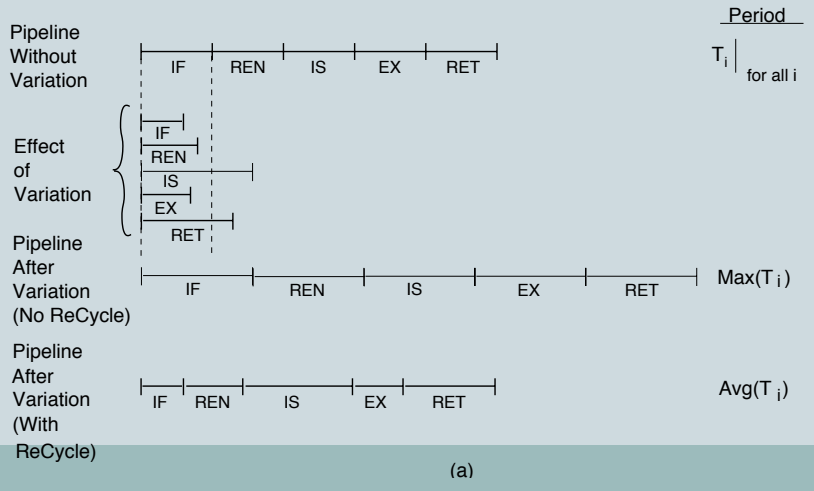
- Process Variation
  - As transistor sizes continue to shrink, our their error margins continue to become more significant
    - The same stage on two different chips may not be equal with regard to timing
    - This requires designers to set more conservative cycle times, which will in turn affect the cycle time of all stages
      - Guard Banding – Bad for performance!
- Also, like Razor, Power
  - Same arguments as Razor…
- Finally, make average silicon matter!
  - Salvage chips that vary beyond the set threshold (and therefore fail hold-time tests)

ReCycle – Approach

- What's unique to ReCycle?
  - Analyzing feedback paths in cycle time analysis
  - Using cycle time stealing to tackle process variation
  - The implementation of Donor stages, but not the idea in its entirety
- What did it inherit?
  - The notion of skewing the clock to alter cycle times
  - Mapping the clock skew optimization problem as a graph

ReCycle – Approach

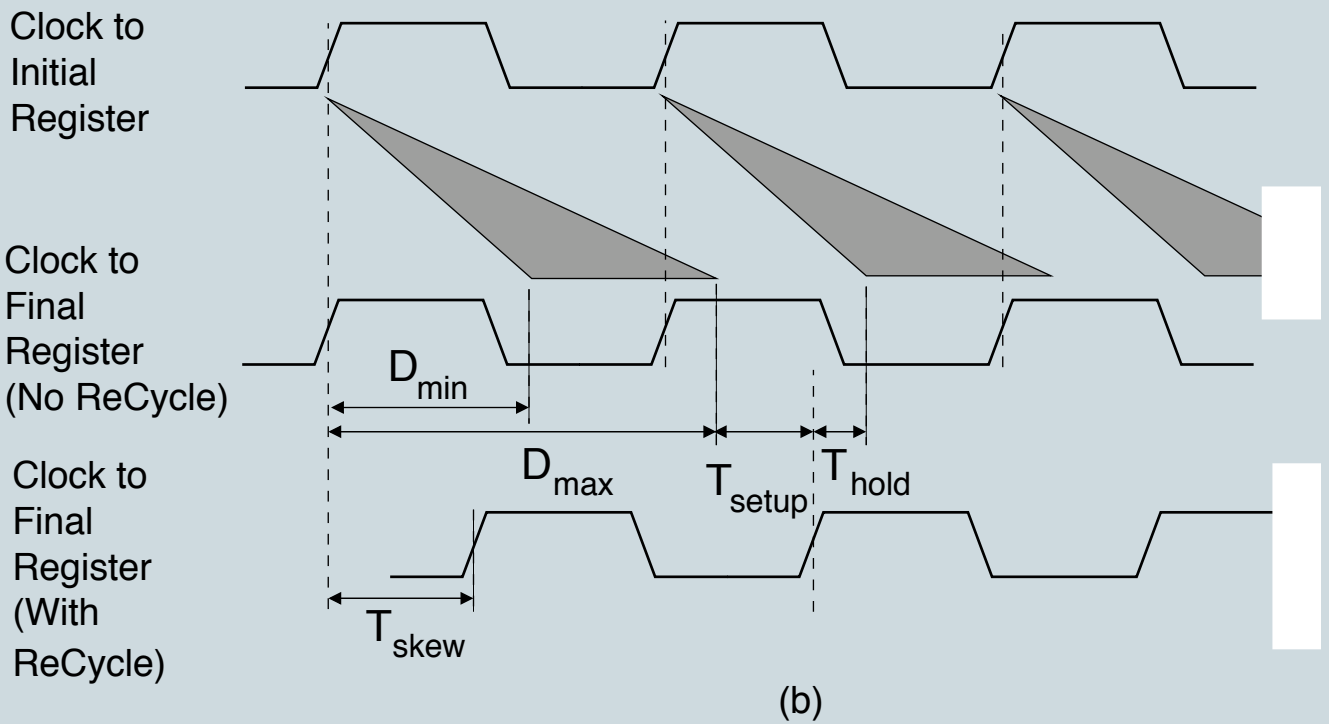
- The Main Idea
  - Why should we increase the cycle time of all of a pipeline's stages if only one or two stages are the culprits of long cycle times (due to variation)?
    - This greater level of unbalance is actually more optimal for ReCycle
  - Why doesn't a designer concentrate on the average cycle time instead of the longest?
    - Let one slower stage "borrow" or "steal" time from a faster stage



(a)

ReCycle – Approach

- Illustration of how ReCycle skews the clock to account for process variation:



ReCycle – Uses

- Long Pipelines
  - ReCycle continues to outperform a non-ReCycle scheme at an exponential rate as the number of stages are added to a given pipeline
- Donor Stages
  - Can increase the frequency of a pipeline by adding empty “donor” stages.
  - With ReCycle, this essentially behaves in a similar way to adding a pipeline stage and rebalancing the stages to fit equally in the new pipeline slots.
  - This can be done either statically or dynamically
    - Statically-Donor Algorithm is run a single time on each new chip
    - Dynamically-Donor Algorithm is run on a “new phase” (as seen by a phase detector)

ReCycle – Uses

- ReCycling to Feedback Paths
  - Remembering back to the *Future of Wires* paper, longer wires—in this case, feedback paths—make use of repeaters to break the quadratic relation between wire delay and wire length:
    - Delay = constant*wire_length²
  - Non-critical pipeline loops will redirect excess cycle time (slack) to reduce overall repeater usage:
    - In this case, the slack is used to reduce a feedback path's requirement for speed, thereby reducing its dependence on frequent repeaters.

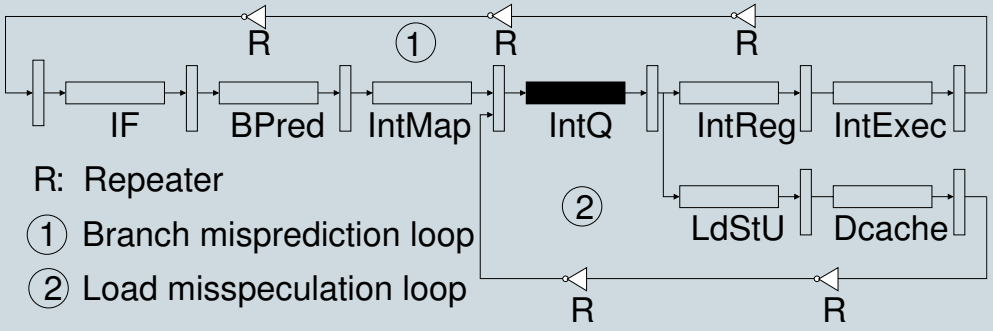


Figure 4: Example of overlapping loops.

ReCycle - Uses

- More on Feedback Path ReCycling
  - By reducing the number of repeaters, we are also saving a great deal of power.
  - Research also shows that a reduction in repeaters could be very helpful for future power reductions

Table 2: Comparison of no. of repeaters of our approach with previous work. The numbers shown are for 70 nm technology

Number of repeaters estimated by [4]	Number of repeaters Estimated by [15]	Our approach, p=0.55	Our approach, p=0.6
1.6 million	0.2 million	0.85 million	1.61 million

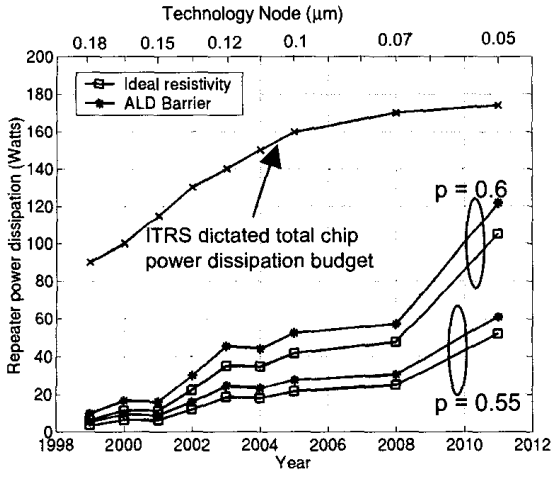


Figure 5: Repeater power dissipation as function of tech node. ITRS dictated total chip power budget also shown.

ReCycle - Uses

- More on Feedback Path ReCycling
  - Also, with more cycle time for wires versus repeaters, a designer is allowed more freedom with routing.
- Catering to the Average Case
  - Designs equipped with ReCycle will also have the ability to correct hold violations post-fabrication.
    - Greater yield -> Lower prices

ReCycle – Implementation

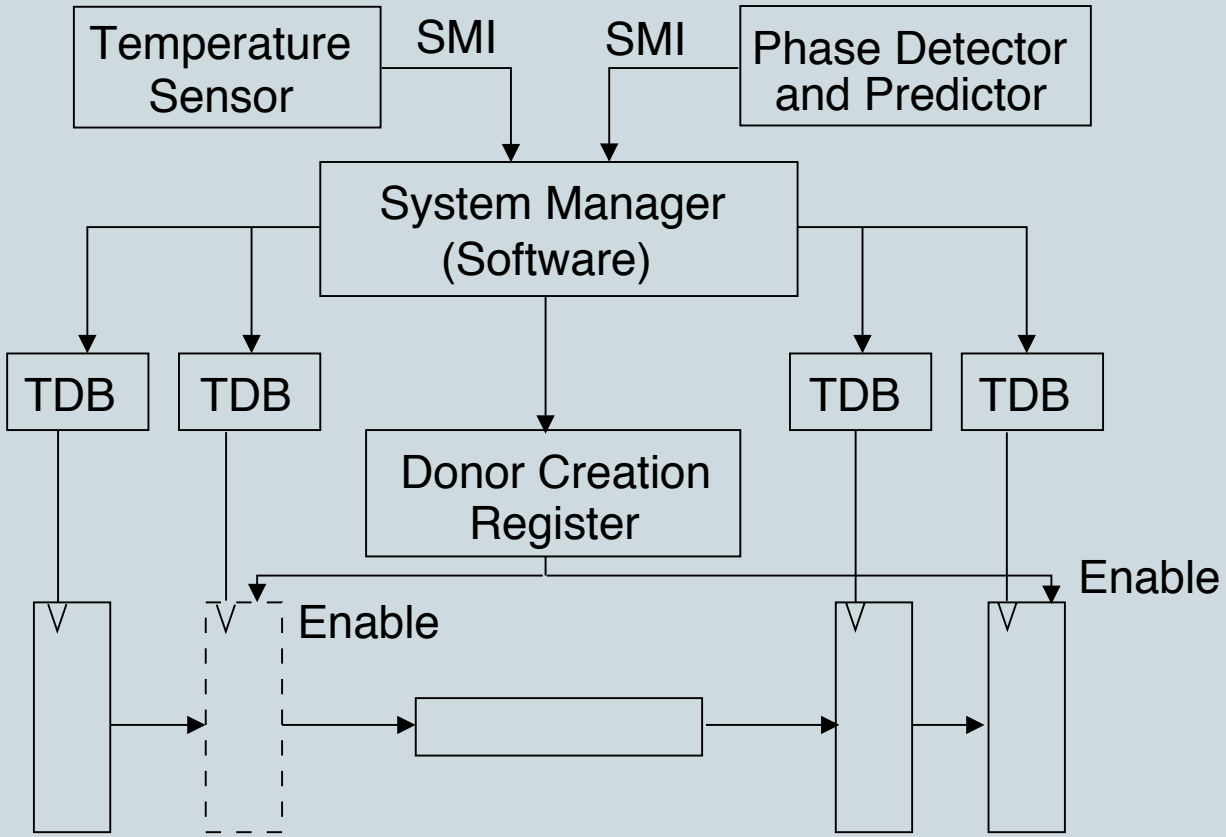


Figure 6: Overall ReCycle system.

ReCycle – Simulation/Results

- Simulation Model
  - Alpha 21264-based
  - 64 KB L1 I/D Caches
  - 2MB L2 Cache
  - Balanced Pipeline Stages
  - 45 nm Feedback wire proces

ReCycle – Results

- ReCycle is able to reclaim almost 60% of the frequency lost to process variation.
  - The simulation was fixed at a useful logic depth per stage of 17FO4 (measure of delay)

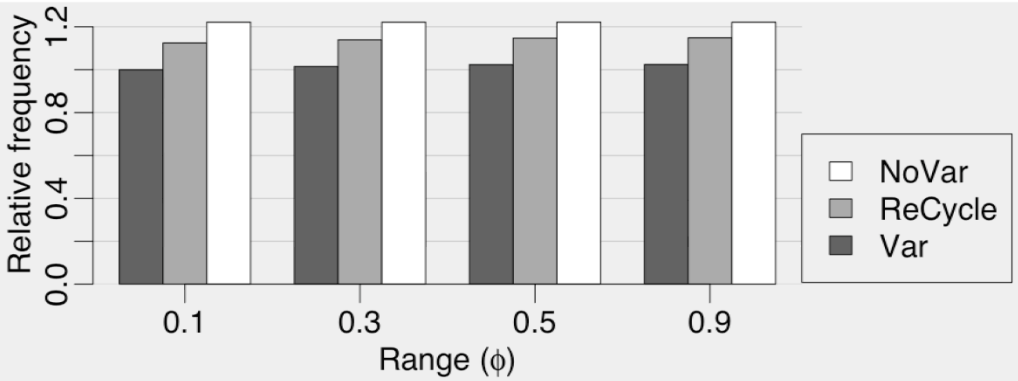


Figure 10: Pipeline frequency of the environments considered for different ϕ.

ReCycle – Results

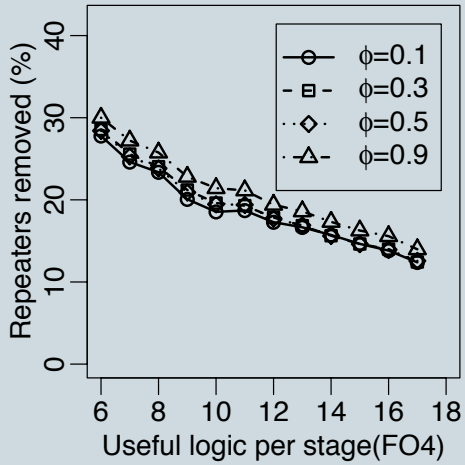


Figure 16: Fraction of repeaters eliminated by ReCycle.

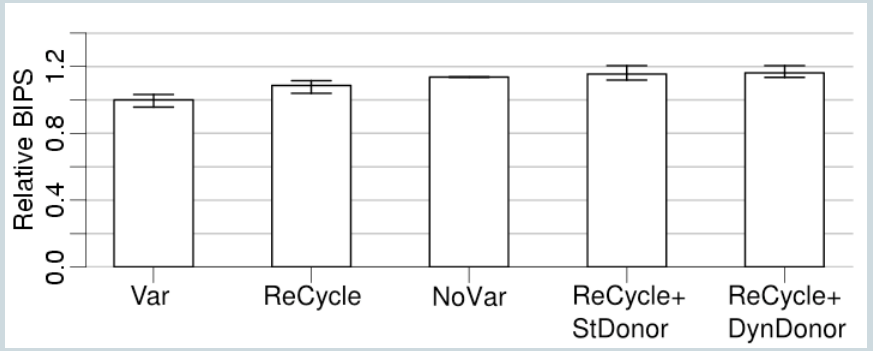


Figure 14: Performance of different environments.

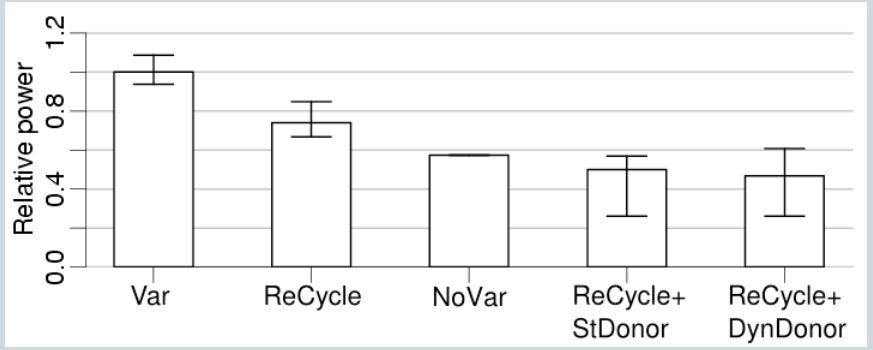


Figure 15: Dynamic power for constant performance.