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

Matthew Fojtik, David Fick, Yejoong Kim, Nathaniel Pinckney, David Harris, David Blaauw, Dennis Sylvester
mfojtik@umich.edu

Electrical Engineering & Computer Science Department
The University of Michigan, Ann Arbor
Outline

- Issues with Prior Razor
- Bubble Razor Algorithm
- Circuitry and Implementation
- Area Overhead Tradeoffs
- Test Chip Results
Timing Margins

Margins for uncertainty:
- Process Variation
- Temperature Variation
- Voltage Variation
- Aging Effects

Associated Costs:
- Lost performance
- Lost energy
- Tester time (tradeoff)
Eliminating Margins

- Always Correct
  - Tables, Canaries

- Detect and Correct
  - Razor Style

<table>
<thead>
<tr>
<th>Technique</th>
<th>Process</th>
<th>Ambient</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Global</td>
<td>Local</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Slow</td>
</tr>
<tr>
<td>Table Lookup</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Table &amp; Sensors</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Canary Circuit</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Razor Designs</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

S. Das, et. al. [VLSI 2005]
Speculation Window and Hold Time

Speculation window linked to minimum delay constraint (hold time)
Architectural Invasiveness

Razor I Style – All Flops Reload Previous Values

Razor II Style – Check Stage and Architectural Replay

- Requires Designer Effort
- RTL written with Razor in mind
Fundamentals of Bubble Razor

- Two-Phase Latch Timing
  - Automatically convert Flip-Flop based design

- Time Borrowing as Correction Mechanism
  - Does not modify design architecture
  - Does not require reloading / replaying instructions

- Local Correction (Bubbles)
  - Break requirement of stalling entire chip at once
Two Phase Latch Razor Timing

Larger Speculation Window

Minimum delay constraint the same as conventional design
Time Borrowing as Error Correction

Bubble Razor – Switch to Latches, Borrow Time

- No Hold Time Issues
- Architecture Agnostic
- Push-button approach
- No metastability on datapath
Stalling Locally with Bubbles

Stalling the Clock Locally
• With flops, all registers hold data
• With latches, half registers hold bubbles
• Every latch stalls exactly once
• Communication only between neighbors

Eventually it all resolves

Blue tells Green to stall
Purple tells Blue to stall
Red tells Purple to stall
Yellow tells Red to stall
Yellow tells downstream no new data exists
Yellow stalls Not immediately overwritten

Time
Timing of Clock Waveforms

1. Should Arrive
2. Timing Violation
3. Give Time to Recover
4. Prevent Double Sampling inst1
5. Prevent Losing inst2
6. Prevent Losing inst3
Timing of Clock Waveforms

1. Should arrive
2. Timing violation
3. Give time to Recover
4. Prevent Double Sampling inst1
5. Prevent Losing inst2
6. Prevent Losing inst3
7. Prevent
8. inst2
9. inst3
10.
Timing of Clock Waveforms

Diagram showing the timing of clock waveforms with numbered elements 1 to 10.
Timing of Clock Waveforms

Timing violation

Stall Neighbors

Stall 3
The Required Circuitry
Error Detection And OR Circuitry

Bubble Razor Latch

Main Latch

1
A cluster stalls and sends bubbles to all neighbors if:
- Told by a neighboring cluster
- Did not stall in the previous cycle
- Equivalent to sending bubbles to “other” neighbors
Clustering with hMETIS

- Widely used Hypergraph partitioning program, hMETIS
- Clusters must only contain members with the same phase
  - Create two graphs, and partition independently
- Connected in hMETIS graph, if transitively connected in circuit
  - Edge Weight = number of latches that form transitive connection
Clustering Results

- Tradeoff between sizes of OR gates
- Combining errors
- Combining bubbles

- 100 negative clusters
- 70 positive clusters
Two Port Memory Boundary Approach

Must fit edge triggered memory into stalling algorithm
“Managing” the Synthesis/APR Tools

- Want balanced pipelines, no time borrowing
  - Model razor latches as flip flops
- Dynamic OR always followed by latch
  - Model dynamic OR as static
  - Model latch as flip flop (captures when latch closes)
- Use regular ICG cells
  - Can use conventional clock tree synthesis
- Final design appears to be relatively “normal”
  - Flip-flop based design with clock gating
  - Everything is timing constrained
- “Razorization” process is entirely automated
  - Synthesis and netlist transformation scripts
Retiming can increase the number of latches
Results in area overhead
Area Overhead of Latch Transformation

- **Flip-Flop Based Cortex-M3**
- **Latch Based Cortex-M3**

Graph showing the total area (normalized) vs. clock frequency (normalized) for both flip-flop and latch-based Cortex-M3 configurations. The graph indicates a 7% and 8% overhead for flip-flop and latch-based configurations, respectively, at certain clock frequencies.
Speculation Window Size

- Full Clock Phase (100%) Minus Delay of Error Propagation Circuits
  - Maximum allowed by technique

- Number / Location of Latches with Error Checking
  - Maximum slowdown that does not result in unchecked error
Where Error Checking is Needed

- If circuit delay suddenly becomes 130% of its nominal value, all timing errors will be detected before the circuit fails.
Path Distribution for Cortex-M3

Flip Flops

Negative Latches

Positive Latches

All Latches
Area Increase from Error Checking

- Checking All Latches
- Checking Negative Latches
- Checking Positive Latches

20% Area Overhead
30% Timing Speculation
Implementation on ARM Cortex-M3

### ARM Cortex-M3

**Processor Core**: ARM Cortex-M3

**Process Technology**: IBM 45nm SOI12S0

**Nominal VDD**: 1.0 V

**SRAM Size**: 16 kB

**Latches**: 7159

**Positive Clusters**: 70

**Negative Clusters**: 100

**Speculation Window**: 55%
Characterizing Throughput / Energy

- Operating Point Set for Worst Case Operation
  - 85°C
  - 10% Supply Droop
  - 2σ Process
  - 5% Safety Margin

- 200 MHz at 1.0 V
Gains from Bubble Razor

Throughput (FFT/msec)

- Margined
  - +103%

- First Error
  - +67%
  - +22%

- Optimal
Gains from Bubble Razor

Energy Consumption ($\mu$J/FFT)

- Margined: 3.0
- First Error: 1.6
- Optimal: 0.8

-54% and -62% reductions compared to Margined.
Bubble Razor Results

Slow

Throughput (FFT/msec)

- Margined
- First Error
- Optimal

Energy Consumption (μJ/FFT)

- Margined
- First Error
- Optimal

Average

Throughput (FFT/msec)

- Margined
- First Error
- Optimal

Energy Consumption (μJ/FFT)

- Margined
- First Error
- Optimal

Fast

Throughput (FFT/msec)

- Margined
- First Error
- Optimal

Energy Consumption (μJ/FFT)

- Margined
- First Error
- Optimal

Slow: +85%, +45%, +28%

Average: +103%, +67%, +22%

Fast: +112%, +82%, +17%

Margined: -43%, -20%, -54%

First Error: -54%, -16%, -62%

Optimal: -60%, -14%, -66%
# Bubble Razor Results

<table>
<thead>
<tr>
<th>Worst Case</th>
<th>200 MHz</th>
<th>8.5 FFT/ms</th>
<th>1.0 V</th>
<th>3.08 μJ/FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Failure</td>
<td>333 MHz</td>
<td>14.2 FFT/ms</td>
<td>0.775 V</td>
<td>1.42 μJ/FFT</td>
</tr>
<tr>
<td>Optimum</td>
<td>425 MHz</td>
<td>17.3 FFT/ms</td>
<td>0.725 V</td>
<td>1.18 μJ/FFT</td>
</tr>
</tbody>
</table>
Conclusion

- First Razor style implementation on a complete, commercial processor (ARM Cortex-M3).
- Proposed two-phase latch based Razor technique
- Novel local replay algorithm
- Demonstrated automated nature of technique
- Successfully implemented and fabricated in 45nm
- 60% energy efficiency or 100% throughput increase over worst case margining