LECTURE 2: MANAGING VARIATIONS THROUGH ADAPTIVE COMPUTING

Shidhartha Das
Principal Engineer, ARM R&D, U.K

MEDIAN TRAINING SCHOOL, 2013
Outline

- In situ techniques
- “Let Fail and Correct/Error-Tolerant” Adaptive Techniques
- Techniques for General Purpose Computing – Razor
- Brief Introduction to Razor
  - SEU Tolerance through Razor
- ARM ISA based Razor Processor
  - Live Demo with showing Razor based adaptation
  - Parametric yield improvement using Razor
  - Intel EDS
- Techniques for Signal Processing and Communication
- Examples of adaptive techniques similar to Razor
Impact of Local Variations on Tracking Circuits

Rising local variations at advanced geometries undermine the efficacy of canary circuits

Canary circuits need margins to function

The “first” droop is fast-changing
Can be highly localized
Requires *in situ* tracking
Always Correct Approaches – In-situ Method

- In-situ delay detection methods [Kehl ’93]
  - Sample data multiple times in clock cycle

- Eliminates tracking issue and addresses local variations
- Only slow rate changes
- System must be taken off-line and calibrated
Triple-latch Monitor: Frequency Tuning Strategy

- **Compare samples with data**

- **Case I**
  - Data transitions before all edges
  - EQ0, EQ1, EQ2 are all low
  - System running too slow

- **Case II**
  - Data transitions after second edge
  - EQ0 and EQ1 are high
  - System is running too fast

- **Case III**
  - Data transitions after first edge
  - EQ0 is high
  - System is tuned
Triple-latch Monitor Based Self-tuning Circuit

- In situ delay measurement eliminates margins for *global and local* PVT variations

- Periodically, requires critical path vectors to tune the circuits
  - Processor needs to be stopped, tuned and restarted again to set frequency ceiling
  - Can not tune for data-dependent delay variations
    - Sensitizing all critical paths may be difficult

- Worst-case safety margins required for fast transient variations
  - Fast supply voltage variations – RI and Ldl/dt
  - Delay variations due to noise

- Small inherent margin in delay detector
“Error-Tolerant” Approaches

Key Observation: worst-case conditions highly improbable

- Many sources of variability are independent (process, noise, SEU, VDD)
- Probability of all sources simultaneously having worst-case condition very low

Common case design paradigm

- Significant gain for circuits optimized for common case

Efficiency mechanisms needed to tolerate infrequent worst-case scenarios

- In-situ error detection and correction
- Dynamic runtime adjustment to silicon and environmental conditions

Error-tolerance for Signal-Processing and Communications

- ANT, Self-Calibrating Interconnects

Error-tolerance for Microprocessors (Focus of this talk)

- Razor, Intel EDS
Elimination of Voltage Margins with Razor

**Key idea:** Exploit the dynamic nature of variations

- Speculatively operate without full setup margin
- Explicitly check for late-arriving signals
- In the event of a timing error, invoke system recovery mechanism
- Adapt VDD/CLK to target near-zero error-rate operation

![Diagram showing energy and IPC against voltage and recovery energy. Traditional Margin Point and Point of First Failure are indicated. The graph is analogous to wireless communication.]
Razor principles

**Survive** fast moving and transient changes
- Ldi/dt
- Localized IR drop
- PLL Jitter
- Capacitive coupling
- Critical-path sensitization (data-dependent margins)

**Adapt** to slower moving or static conditions
- Global or long-term IR drop
- Low-frequency supply ripple
- Temperature
- Ageing
- Process Variation
Gains from In-situ Error Detection

- **Power with worst-case margins**
  - Safety margin (Uncertainty in models, calibrations)
  - Dynamic Margins (V, T, Noise)
  - Static Margins Process + ageing

- **Power after Razor enabled margin elimination**

- **Power after Razor enabled margin elimination and sub-critical operation**
  - Margin-elimination
  - Sub-critical operation

*Point of First Failure*

- **Power @ PoFF**
- **Power @ PoFF**
- **Power @ low-error rates**
**Razor I: Error Detection Scheme**

Augment flip-flops on critical path with a *shadow* latch which samples off the negative clock edge.

Upon failure: Overwrite main f/f with correct data from the shadow latch

- Ensure that the shadow latch is always correct by conventional design

**Key design issues:**
- Recovering pipeline state while maintaining forward progress after errors
- Meeting hold time constraint on the shadow latch using delay buffers
- Suitably detecting metastability at the main flip-flop
Razor DVS: Pipeline Recovery Scheme

- Multiple cycle penalty for timing failure
- Scalable design since all recovery communication is local
Lessons from the RazorI prototype

- **Safety margin (Uncertainty in models, calibrations)**

- **Dynamic Margins (V, T, Noise)**
  - Power after Razor enabled margin elimination ~35%

- **Static Margins Process + ageing**

- **Sub-critical operation**
  - Power saving through sub-critical operation ~10%

- **Does not scale to larger designs with critical clock-gating enables**

- **Most of energy savings realized by margin elimination**

- **Error-rates are very low at PoFF but increase exponentially, thereafter**

![Graph showing percentage error rate vs voltage](Chip_1.png)
RazorII

- 64-bit 7-stage Alpha processor 0.13µm
- Error detection in Razor FF
- Error correction by architectural replay
- Naturally detects SER without overhead

Source: Blaauw et al. [ISSCC 2008], Das et al. [JSSC 2009]
Razor-enabled energy-efficient ARM processor

**UMC 65SP Process**
- 1V nominal VDD and 1.1V Overdrive

**Implements a sub-set of ARM ISA**
- Critical-paths representative of ARM industrial processor designs

**87 die from split lots**
- 30FF/37TT/20SS

**724MHz sign-off frequency**
- 0.9V/SS/125C

**Adaptive Control Experiments**
- Adaptive Frequency Control - DFS
- Adaptive Voltage Control - DVS
ARM Razor

- Razor Flip-flop protects critical-path endpoints and includes an extra latch for diagnostics purposes
- Standard 5-stage pipeline modified to support recovery by architectural roll-back
- Recovery control interfaces to a on-chip DFS controller and on-board DVS controller for error-rate based adaptive control
Live Demonstration Showing Razor-based Dynamic Adaptation
How does a 700MHz sign-off turn into a 1+ GHz chip?
Scenario A: sign-off at higher voltage

- Slowest chip runs worst-case code at 1.17V (unmargined)
- Success: all chips can reach 1GHz at 1.2V!
- … but sustained operation above 1.1V causes excessive leakage and reduces chip life-time, therefore this is not really a solution
Scenario A.1: prune distribution

1. STA is 724 MHz @ 0.9V
2. Turn up voltage until all run at 1GHz
3. Discard parts that require more than 1.1V
   - Because >1.1V degrades chip life-time

What does this mean in practice?
- Discard all SS parts
- Discard 33% of TT parts
- Pray for some fast parts…

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TT5</td>
<td>1.061</td>
<td>Pass</td>
</tr>
<tr>
<td>TT7</td>
<td>1.062</td>
<td>Pass</td>
</tr>
<tr>
<td>TT19</td>
<td>1.065</td>
<td>Pass</td>
</tr>
<tr>
<td>TT17</td>
<td>1.068</td>
<td>Pass</td>
</tr>
<tr>
<td>TT8</td>
<td>1.068</td>
<td>Pass</td>
</tr>
<tr>
<td>TT9</td>
<td>1.071</td>
<td>Pass</td>
</tr>
<tr>
<td>TT31</td>
<td>1.071</td>
<td>Pass</td>
</tr>
<tr>
<td>TT47</td>
<td>1.072</td>
<td>Pass</td>
</tr>
<tr>
<td>TT34</td>
<td>1.079</td>
<td>Pass</td>
</tr>
<tr>
<td>TT3</td>
<td>1.08</td>
<td>Pass</td>
</tr>
<tr>
<td>TT18</td>
<td>1.08</td>
<td>Pass</td>
</tr>
<tr>
<td>TT32</td>
<td>1.084</td>
<td>Pass</td>
</tr>
<tr>
<td>TT16</td>
<td>1.084</td>
<td>Pass</td>
</tr>
<tr>
<td>TT10</td>
<td>1.087</td>
<td>Pass</td>
</tr>
<tr>
<td>TT33</td>
<td>1.09</td>
<td>Pass</td>
</tr>
<tr>
<td>TT45</td>
<td>1.09</td>
<td>Pass</td>
</tr>
<tr>
<td>TT30</td>
<td>1.102</td>
<td>Fail</td>
</tr>
<tr>
<td>TT15</td>
<td>1.11</td>
<td>Fail</td>
</tr>
<tr>
<td>TT26</td>
<td>1.114</td>
<td>Fail</td>
</tr>
<tr>
<td>TT2</td>
<td>1.122</td>
<td>Fail</td>
</tr>
<tr>
<td>TT27</td>
<td>1.126</td>
<td>Fail</td>
</tr>
<tr>
<td>TT28</td>
<td>1.144</td>
<td>Fail</td>
</tr>
</tbody>
</table>
Scenario R: use Razor
Parametric Yield – Native Distribution

87 devices at 1.1V

Power at 1GHz (mW)

Maximum Frequency at 1.1V (MHz)

Number of Chips
Parametric Yield – Power vs Frequency

- **Power at 1GHz (mW)**
- **Maximum Frequency at 1.1V (MHz)**

- 87 devices at 1.1V
- 1.1V OD
- FF5
- SS6
- FF (30)
- TT (37)
- SS (20)
Parametric Yield – Power vs Frequency

87 devices at 1.1V

Power at 1GHz (mW)

Maximum Frequency at 1.1V (MHz)

Power Limit

1.1V OD

Frequency Limit

SS6

FF5

FF (30)

TT (37)

SS (20)
Parametric Yield – Prune Distribution

- **Power Limit**
- **1.1V OD**
- **Yielding Parts**
- **Frequency Limit**

---

- **Power at 1GHz (mW)**
- **Maximum Frequency at 1.1V (MHz)**
Parametric Yield – Prune Distribution

Power at 1GHz (mW)

1.1V OD

>60mW (21)

Yielding Parts

= 28 out of 87

<1GHz (38)

Maximum Frequency at 1.1V (MHz)
Parametric Yield – Razor

- Power at 1GHz (mW) > 60mW (0)
- Yielding Parts = 87 out of 87
- Maximum Frequency at 1.1V (MHz)
Error-Detecting Sequentials

- EDS assigned during synthesis convergence and embedded in critical-paths
- About 12% of total sequentials are EDS
Error-Detecting Sequential

- Tunable Replica Circuits (TRC) separately monitor the critical-paths and are tuned through explicit calibration
- TRC failure initiates recovery and must always fail if critical-path fails
About 16% gains with EDS and 12% gains through TRC
Measured TP Versus FCLK

- About 16% gains with EDS and 12% gains through TRC
Application Dependence

- TRC follows critical-path and hence gains remain fixed at 12%.
- However, EDS is able to take advantage of data-path activation and therefore sees greater gains between 16-20%.
Self-Calibrating Interconnects

- Data is encoded before transmission
- Decoded code-word determines if transmission was successful or not
- Controller adjusts $V_{\text{comm}}$ and $F_{\text{comm}}$ according to error-rates observed at decoder

Partial reuse with permission from Worm et. al., [EPFL 2002]
Main Block is designed for average case
- Makes intermittent errors
Estimator approximates Main Block output based on last output
Compare (error-detection) and replace (error-correction)
Conclusion

- Error-tolerant approaches
  - Remove additional margins but increase design complexity
  - Can take advantage of signal processing applications with fixed BER
- Error-tolerant computing in some cases also translates into approximate computing where applications do not require bit-exact outputs
  - However, such techniques are not adaptive
- Traditional worst-case computing model is being increasingly replaced with adaptive techniques even in commercial applications
- However, there are design complexities involved
  - Always-correct techniques require significant calibration and characterization efforts
  - Error-tolerant techniques require significant validation/verification efforts