Better Than Worst-Case Design

Todd Austin
University of Michigan
austin@umich.edu

Acknowledgements:
Valeria Bertacco, David Blaauw, Shidhartha Das, Dan Ernst, Nam Sung Kim, Seokwoo Lee, Trevor Mudge, Chris Weaver
Kris Flautner & ARM Ltd
Robert Colwell

Challenges in the Nanometer Regime

- Design complexity
  - Billions and billions of transistors lead to untenable designs...
- Device-level faults in logic and memory
  - Cosmic rays, alpha particles, gate wear-out, silicon defects, etc...
- Uncertainty in design parameters
  - Process and temperature variation, supply noise...
- Power/performance demands
  - Bounding performance, area, and battery life
Traditional Worst-Case Design

Better Than Worst-Case Design
**Presentation Agenda**

- BTWC Design Examples
  - DIVA Checker
  - Razor Logic
- BTWC Design Opportunities and Challenges
  - Typical-Case design Optimization (TCO)
- Conclusion

**Example BTWC Design: DIVA Checker [MICRO ‘99]**

- All core function is validated by checker
  - Simple checker detects and corrects faulty results, restarts core
- Checker relaxes burden of correctness on core processor
  - Tolerates design errors, electrical faults, defects, and failures
  - Core has burden of accurate prediction, as checker is 15x slower
- Core does heavy lifting, removes hazards that slow checker
Checker Processor Architecture

Check Mode
Recovery Mode

How Can the Simple Checker Keep Up?

- Slipstream reduces power requirements of trailing car
- Checker processor executes inside core processor’s slipstream
  - fast moving air ⇒ branch predictions and cache prefetches
  - Core processor slipstream reduces complexity requirements of checker
  - Checker rarely sees branch mispredictions, data hazards, or cache misses
**How Can the Simple Checker Keep Up?**

- Slipstream reduces power requirements of trailing car
- Checker processor executes inside core processor’s slipstream
  - fast moving air ⇒ branch predictions and cache prefetches
  - Core processor slipstream reduces complexity requirements of checker
  - Checker rarely sees branch mispredictions, data hazards, or cache misses
REMORA: Physical Checker Design

- Physical checker design
  - Alpha integer ISA subset
  - 4-wide checker, 0.5k I-cache, 4k D-cache
- Less than 3% slowdown for Alpha core
- Only a 6% area overhead incurred
- Design also includes:
  - Pipelined checker design, simple core
  - Clock/voltage tuning infrastructure
  - Extensive BIST support

Alpha 21264

REMORA Checker

12 mm² (in 0.25µm)

205 mm² (in 0.25µm)

Verifying the Checker Processor

- Simple checker permits complete functional verification
  - In-order blocking pipelines (trivial scheduler, no rename/reorder/commit)
  - No "internal" non-architected state
- Fully verified design using Sakallah’s GRASP SAT-solver
  - For Alpha integer ISA without exceptions
  - With small register file and memory, and small data types

Unspecified Core Predictions

Checker Model

Reference Model (ISA sim)

X

Always true if uArch model == Ref model

Identical state?
Presentation Agenda

- BTWC Design Examples
  - DIVA Checker
  - Razor Logic
- BTWC Design Opportunities and Challenges
  - Typical-Case design Optimization (TCO)
- Conclusion

Motivating Study:
Voltage vs. Circuit Error Rate
Circuit Under Test

Error Rate Studies – Empirical Results

35% energy savings with 1.3% error
22% saving

Supply Voltage (V) once every 20 seconds!
**Error Rate Studies – SPICE-Level Simulations**

- Based on a SPICE-level simulations of a Kogge-Stone adder

![Graph showing error rate studies](image)

**Another BTWC Design: Razor Logic**

- Double-sampling latches detect timing errors
  - Second sample is correct-by-design
- Microarchitectural support restores state
  - Timing errors treated like branch mispredictions
- Research challenges: metastability and short-path constraints
Distributed Pipeline Recovery

- Builds on existing branch prediction framework
- Multiple cycle penalty for timing failure
- Scalable design as all communication is local

Razor Prototype

- Dimensions: 3.3mm x 3.0mm
- Includes Icache, Dcache, IF, ID, EX, MEM, WB blocks
Razor Opportunity: Typical-Case Energy Reduction

- Energy reduction can be realized with a simple proportional control function
  - Control algorithm implemented in software

Voltage Controller Response

- Two minute snapshot of a 15 min run
Energy/Performance Characteristics

Energy of Processor Operations, $E_{\text{proc}}$

Energy of Pipeline Recovery, $E_{\text{recovery}}$

Total Energy, $E_{\text{total}} = E_{\text{proc}} + E_{\text{recovery}}$

Optimal $E_{\text{total}}$

50%

Measured Results

Voltage at First Failure

Voltage at 0.1% Error Rate

Chips

Point of 0.1% Error Rate Vs Point of First Failure

Linear Fit $y = 0.78685x + 0.22117$

Normalized Energy Savings over First Failure Point at 0.1% Error Rate
Other Better Than Worst-Case designs

- Algorithmic-Noise Tolerance, Shanbhag et al.
  - Converting circuit faults to S/N component
- Approximate Circuits, Lu et al.
  - Architecture-level speculation on computation
- TEAtime Adaptive Clock, Uht et al.
  - Adaptive clock control
- On-Chip Self-Calibrating Busses, Worm et al.
  - Error recovery logic for on-chip busses
- Self-Tuning Circuits, Kehl et al.
  - Early work on dynamic timing error avoidance
- Time Based Transient Fault Detection, Anghel et al.
  - Double sampling latches for speed testing

March 2004

Presentation Agenda

- BTWC Design Examples
  - DIVA Checker
  - Razor Logic
- BTWC Design Opportunities and Challenges
  - Typical-Case design Optimization (TCO)
- Conclusion
**BTWC Design Opportunities**

- **Key observation:**
  
  *Infrequent faults in the core design are tolerable.*

- **Opportunities:**
  
  - Focus only on the critical components, no need to verify *ad infinitum*
  - Optimize performance/power/implementation for the most common scenarios *(typical-case optimization)*

---

**BTWC Design Opportunity: Typical-Case Optimized Adder**

Kogge-Stone Adder
Carry Propagations for Random Data

Carry Propagations for Typical Data
**Typical Case Optimized Adder**

- **Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12, Q13, Q14, Q15, Cin**
- **G0, P0, G1, P1, G2, P2, G3, P3, G4, P4, G5, P5, G6, P6, G7, P7, G8, P8, G9, P9, G10, P10, G11, P11, G12, P12, G13, P13, G14, P14, G15**
- **Cout**

---

**Benefits of Typical Case Optimization**

<table>
<thead>
<tr>
<th>Adder Topology</th>
<th>Latency (in gate delays)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Worst-Case</td>
</tr>
<tr>
<td>Kogge-Stone</td>
<td>8</td>
</tr>
<tr>
<td>TCO Adder</td>
<td>16</td>
</tr>
</tbody>
</table>

- **Typical-case performance much better than worst case**
  - Especially for typical-case optimized design
**Presentation Agenda**

- BTWC Design Examples
  - DIVA Checker
  - Razor Logic
- BTWC Design Opportunities and Challenges
  - Typical-Case design Optimization (TCO)
  - Circuit-level observability and system-level performance
- Conclusion

**BTWC Design Challenge: Observability of Circuit-Level Characteristics**

- Circuit-Aware Architectural Simulator efficiently melds circuit simulation with architectural simulation
Conclusion

- Better than worst-case design abandons traditional worst-case design constraints
- Couples complex designs with checkers
  - DIVA Checker verifies program computation
  - Razor Logic verifies circuit timing
- Enables CAD opportunities for typical-case optimization