Built-In Soft Error Resilience for Robust System Design

Subhasish Mitra

Intel Corporation
Folsom, CA

(Full-time Stanford faculty starting 01/06)

Email: smitra@crc.stanford.edu
Acknowledgment

- John Crawford, Kiran Ganesh, Kee Sup Kim, John Madrid, Jose Maiz, TM Mak, Bijan Mayelzadeh, Stefan Rusu, Norbert Seifert, Quan Shi, Paul Shipley, Nick Vocke, Saad Waqas, Jiu-Shang Yang, Ming Zhang (ex-intern), Victor Zia, & several others of Intel
Robust Computing System

Inputs:
- Defects, Process variation, Degraded transistors
- Radiation, Noise
- Design errors, Software failures
- Malicious attacks, Human errors

“Acceptable” Outputs:
- Performance
- Power
- Data Integrity
- Availability
- Security
Robust Design Paradigm – Avoidance

- Conservative design
- Design validation
- Thorough test
  - Hardware
  - Software
- Infant mortality screen
- Transient error avoidance

Several challenges in future technologies
Robust Design Paradigm – Tolerance

- Error detection during system operation
- On-line diagnostics
- Attack prevention
- Self-recovery
- Self-repair

Classical duplication very expensive
Classical duplication inadequate
Automation for optimized protection required
Previous Research at Intel: X-Compact + Xpand

- Massive reduction
  - Scan test data, test time
- > 40 products
- Highest Intel award recipient

Novel DFT schemes get real results

By Nicolas Mokhoff
EE Times
October 13, 2003 (11:03 a.m. ET)

Charlotte, N.C. - Novel techniques aimed at speeding up the testing of complex computer chips grabbed the spotlight at the International Test Conference in Charlotte, N.C., sponsored by Semicore Technology.
Previous Stanford Research

- Design diversity for error detection
  - Quantitative metric
  - Synthesis for diversity
- Diverse data & duplicated instructions (ARGOS satellite)
- Self-repairing FPGA system (ROAR project)
- Designware synthesis (CAD)
  - Incorporated into Ambit (Cadence) tool
Outline

- Introduction
- Soft error modeling challenges
- Built-In Soft Error Resilience (BISER)
- Conclusion
What Are Soft Errors?

- Transient errors
  - Single Event Upsets, SEU, SER
  - Logic value changed \((0 \rightarrow 1, 1 \rightarrow 0)\)
  - Memory, flip-flops, combinational logic
- Causes
  - Alpha particles from packaging
  - Neutrons from cosmic rays
SRAM soft errors cause hard network problems
Networking equipment is growing increasingly susceptible to soft errors — nonrecoverable, transient errors that can lead to...
Who Cares About Soft Errors?

- “It’s ridiculous. I’ve got a $300,000 server that doesn’t work. The thing should be bullet-proof.”
  - Forbes magazine, 2000

- “SEU exposure far exceeds these server reliability limits.”
  - IBM, Bossen, IRPS 2002

- “All future designs that require highest availability must counter unavoidable SEUs.”
  - Cisco

- “Highest failure rate of all other reliability mechanisms combined.”
  - TI, Baumann, IRPS 2002
Logic Soft Errors

- Soft errors affecting
  - Flip-flops
  - Latches
  - Combinational logic
- Memory soft errors
  - Well understood
    - ECC

Unprotected memory

Soft Error rate contributions
System Effects of Logic Soft Errors

- **Benign**
  - e.g., Dead instructions anyway
  - e.g., Soft errors in network packets
    - Protocol assisted correction – retransmit

- **Silent data corruption**
  - Undetected incorrect results
    - e.g., $ 20,000 interpreted as $ 3,616

- **Detected but uncorrected** – recovery required
Logic Soft Errors & Moore’s Law

- Soft error rate per unprotected chip
  - Roughly double every generation
    - Reason: Double transistor count

<p>| Technology Generation | Undetected Soft Error Rate Per chip | Enterprise Undetected Soft Error Rate Goal | Logic Flip-Flop Protection Required |</p>
<table>
<thead>
<tr>
<th></th>
<th>BISER</th>
<th>Multi-threading</th>
<th>Logic Duplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power penalty (Error resilient)</td>
<td>3 – 4.5%</td>
<td>&gt; 40%</td>
<td>40 - 100%</td>
</tr>
<tr>
<td>Power penalty (Economy)</td>
<td>1.6%</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>Power penalty (Standby)</td>
<td>1.5%</td>
<td>Small</td>
<td>Significant</td>
</tr>
<tr>
<td>Speed penalty</td>
<td>0 – 1.5%</td>
<td>20 - 40%</td>
<td>Almost none</td>
</tr>
<tr>
<td>Area penalty</td>
<td>0.1%</td>
<td>5 - 10%</td>
<td>40 - 100%</td>
</tr>
<tr>
<td>Applicability</td>
<td>Unlimited</td>
<td>Micro-processor</td>
<td>Unlimited</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Logic soft error modeling challenges
  - Latches & flip-flops – Well understood
  - Combinational logic – OPEN
  - System-level effects – OPEN
- Logic soft error protection
- Conclusion
Latches Aren’t Vulnerable All The Time

- Clock input = 1
  - Soft error susceptibility $\approx 0$
- Clock input = 0
  - Soft error susceptibility $\approx 100$
- Overall TVF: 50% (= duty cycle)
- More sophisticated analysis possible
Soft Errors in Combinational Logic

- Logical masking

\[ S = 1 \]
\[ A = 1 \]
\[ B = 1 \]
\[ \text{OUT} = 1 \]
Soft Errors in Combinational Logic

- Electrical masking

\[ S = 1 \]
\[ A = 0 \]
\[ B = 1 \]

Diagram:

- Electrical masking

\[ \text{OUT} \]
\[ 0 \]
Soft Errors in Combinational Logic

- Latching window masking

\[ A = 0 \]
\[ S = 1 \]
\[ B = 1 \]
Architectural Vulnerability Factor (AVF) – aka Logic Derating

- Probability (Given soft error event is BENIGN)
  - No impact on architectural state
- Data registers
  - Average active variable occupancy
- Pipeline flip-flops
  - Average committed instruction occupancy

MAJOR CHALLENGE:
AUTOMATED AVF ESTIMATION
Latch Fault Injection Results

- Similar to Alpha 21164 model, SPEC benchmarks

  State Mismatch + Corrupt Control Word + Timeout

  No Failure

  91%

  9%

  Micro-architectural masking

- Courtesy: Prof. S.J. Patel, UIUC
Outline

- Introduction
- Soft error modeling challenges
- Built-In Soft Error Resilience (BISER)
  - Error blocking
  - Error trapping
  - Results
- Conclusion
Error Protection: Low Hanging Fruits

- Selective node engineering
  - Increased node capacitance
- Soft error rate reduction: 40%
- Costs
  - Power: ~ 3%
  - Performance: None
  - Area: ~ 3%

Selective Node Engineering Data

<table>
<thead>
<tr>
<th>Design block</th>
<th>Soft error reduction</th>
<th>Power overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>38 %</td>
<td>3 %</td>
</tr>
<tr>
<td>Block 2</td>
<td>30 %</td>
<td>3 %</td>
</tr>
</tbody>
</table>
Major Error Protection Techniques

- Circuit hardening – Circuit design literature
- Redundancy – Fault-tolerance literature
  - Duplication, parity codes
  - Multi-threading
  - Software techniques (SIHFT)
- Built In Soft Error Resilience – New technique
- Soft error rate reduction: at least 20X
- Costs: discussed later
Built-In Soft Error Resilience (BISER)

- Soft error resilience
  - Correct flip-flop soft error
    - Error blocking
  - Detect flip-flop soft error
    - Error trapping
- Hardware reuse paradigm
  - Existing on-chip resources reused
    - Design for Test & Debug structures

Ref: IEEE Computer, Feb. 2005
Test and Debug Structure Reuse

- Flip-flop scan reuse
  - Covered today
- Scanout reuse
  - Not covered today
- Pulsed latch scan reuse
  - Not covered today
- MUX scan reuse with Design for Debug
  - Not covered today
Scan Flip-flop Design [ITJ 2004]
System-level Scan Structure

Scan control signals

Scan chain

SDI
SDO
D
Q

Combinational Logic

SDI
SDO
D
Q

Combinational Logic

SDI
SDO
D
Q

Clock
Why is Scan Important?

- Manufacturing test
  - Automated test pattern generation
  - **At-speed** functional test
    - Intel “scanout” approach (aka “sigmode”)
- **At-speed** debug
  - Access to internal nodes of a design
- Diagnosis
  - Manufacturing defects & field failures
Error Resilient Mode

Scan Clock B
Scan Data
Scan Clock A
Capture = 1
Update
System Data
System Clock

1D C1 2D C2
1D C1
1D C1

Scan / Checking Flip-flop
Scan / Duplex Output
System Output
System Flip-flop
C-element

- Extensive use in asynchronous circuit design
Error Blocking Design

Scan Clock B
Scan Data
Scan Clock A
Capture = 1
Update
System Data
System Clock

Scan / Checking Flip-flop

System Flip-flop
System Output
C-element
Keeper
System Output
Error Blocking Operation

Scan / Checking portion

System portion

System Output

Error Blocked
Economy Mode: Core Reuse Enabled

Scan Clock B = 1
Scan Data
Scan Clock A
Capture = 0

Update
System Data
System Clock

Scan / Checking Flip-flop
System Flip-flop
Scan / Duplex Output
System Output

1D C1
2D C2

Q
Q

1D C1
2D C2

Q
Q
Error Blocking Advantages

- > 20X soft error rate reduction
- Minimal overhead
- System-level recovery not required
- Core reuse enabled
Error Detection – Not A Good Idea

- Major issues
  - Global interconnect overhead
  - Error checker area, routing, power overhead
Error Trapping Scan Design

Scan / Checking Flip-flop

Scan Clock B
Scan Data
Scan Clock A
Capture
Update
System Data
System Clock

1D C1 Q
2D C2

1D C1 Q
2D C2

System Flip-flop

Scan Output

XOR

System Output

Scan / Checking Flip-flop
Trapped Error Signal Observation

- Shift out using existing slow scan path
  - At fixed intervals
    - e.g., recovery checkpoint
    - e.g., transaction commit point

😊 No additional global interconnect

😉 Error detection latency
## Library Simulation Results

<table>
<thead>
<tr>
<th></th>
<th>Scan Flip-flop</th>
<th>Error Blocking Scan Flip-flop</th>
<th>Error Trapping Scan Flip-flop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undetected Soft Error Rate</td>
<td>1</td>
<td>&lt; 0.05 – Error Corrected</td>
<td>Almost none</td>
</tr>
<tr>
<td>Power penalty (Error resilient mode)</td>
<td>1</td>
<td>2.13</td>
<td>2.26</td>
</tr>
<tr>
<td>Speed</td>
<td>1</td>
<td>1 – 1.01</td>
<td>1 – 1.01</td>
</tr>
<tr>
<td>Area</td>
<td>1</td>
<td>1.08</td>
<td>1.24</td>
</tr>
<tr>
<td>Global interconnect</td>
<td>NA</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Recovery required</td>
<td>NA</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Lower overheads with Scanout reuse
## Chip-level Results: Error Blocking

<table>
<thead>
<tr>
<th>Undetected soft error Rate reduction</th>
<th>Speed Penalty</th>
<th>Power penalty (Resilient mode)</th>
<th>Power penalty (Economy mode)</th>
<th>Power penalty (Standby Mode)</th>
<th>Area penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x</td>
<td>0 – 1%</td>
<td>0.6%</td>
<td>0.2%</td>
<td>0.2%</td>
<td>0.01%</td>
</tr>
<tr>
<td>4 x</td>
<td>0 – 1%</td>
<td>3%</td>
<td>1%</td>
<td>1%</td>
<td>0.08%</td>
</tr>
<tr>
<td>10 x</td>
<td>0 – 1%</td>
<td>8%</td>
<td>1.5%</td>
<td>1.5%</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

- “Critical” flip-flops protected
- Selected by fault injection on Alpha 21264
- Courtesy: Prof. Sanjay Patel, UIUC
- Lower penalties with “Scanout” reuse
## Comparison with Classical Techniques

<table>
<thead>
<tr>
<th></th>
<th>BISER</th>
<th>Multi-threading</th>
<th>Logic Duplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power penalty</td>
<td>3 – 4.5%</td>
<td>&gt; 40%</td>
<td>40 - 100%</td>
</tr>
<tr>
<td>(Error resilient)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power penalty</td>
<td>1.6%</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>(Economy)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power penalty</td>
<td>1.5%</td>
<td>Small</td>
<td>Significant</td>
</tr>
<tr>
<td>(Standby)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed penalty</td>
<td>0 – 1.5%</td>
<td>20 - 40%</td>
<td>Almost none</td>
</tr>
<tr>
<td>Area penalty</td>
<td>0.1%</td>
<td>5 - 10%</td>
<td>40 - 100%</td>
</tr>
<tr>
<td>Applicability</td>
<td>Unlimited</td>
<td>Micro-processor</td>
<td>Unlimited</td>
</tr>
<tr>
<td></td>
<td>(Core + Uncore)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Soft error modeling challenges
- Built-In Soft Error Resilience (BISER)
- Conclusion
Conclusion

- Robust system design
  - Major research area
- Logic soft errors – extremely important
  - Classical techniques – too costly
- Built-In Soft Error Resilience
  - Efficient & practical
  - Essential for core reuse
Thank You!