Resiliency for Reliability – Myths and Truths

Shekhar Borkar
Intel Corp.
Salishan Conference
April 28, 2015

This research was, in part, funded by the U.S. Government, DOE and DARPA (UHPC, FF1, X-Stack, FF2, CREST). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.
Outline

• Resiliency defined
• Faults, errors, and effects
• Soft-errors
• Permanent faults
• Resiliency framework
• Summary
**Resiliency**

Definition: 
*Asymptotically* provide the reliability of a tri-modular redundancy scheme with only 10% energy and HW cost

State of the art:

<table>
<thead>
<tr>
<th>Technique</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parity, ECC</td>
<td>Memory only, Soft Errors, Erratic bits</td>
</tr>
<tr>
<td>RAZOR</td>
<td>State machines, SER, temporal variations</td>
</tr>
<tr>
<td>Residue logic</td>
<td>Static logic only, permanent faults</td>
</tr>
<tr>
<td>Redundant execution</td>
<td>Memory, RF, SER only</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ \sum \text{Cost} > \text{Cost(tri-modular redundancy)} \] ?

**Resiliency is NOT:**
A solution to error prone shabby engineering!
Resiliency — Three steps

1. Understand faults
   Different types of faults
   Frequency of occurrence, probability, and time to error
   Behavior now, and in the future

2. Understand impact of faults
   Errors caused by the faults (observe)
   Diagnose and pinpoint the fault location
   Recover from the error, correct the fault
   Impact on system performance, energy,…

3. Unified resiliency framework
   Common, serves all types of faults

Cost (Resiliency) << Cost(TMR)
# Understanding Faults

<table>
<thead>
<tr>
<th>Types of Fault</th>
<th>Examples, Effect</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permanent faults</td>
<td>Fan, power supply, shorts and opens</td>
<td>Sensors for detection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Node down</td>
</tr>
<tr>
<td>Gradual spatial faults (Process variations)</td>
<td>Variations in frequency</td>
<td>Design out</td>
</tr>
<tr>
<td></td>
<td>Exacerbated at NTV</td>
<td>Costs perf &amp; energy</td>
</tr>
<tr>
<td>Gradual temporal faults (temp variation with load)</td>
<td>Temperature increase causing frequency loss</td>
<td>Design out</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Costs perf &amp; energy</td>
</tr>
<tr>
<td>Intermittent faults</td>
<td>Data corruption by noise, Soft errors, control loss</td>
<td>Creative accounting</td>
</tr>
<tr>
<td></td>
<td>Not reproducible</td>
<td></td>
</tr>
<tr>
<td>Slow degradation (Aging Faults)</td>
<td>Frequency loss</td>
<td>Design out</td>
</tr>
<tr>
<td></td>
<td>Erratic bits in memory</td>
<td>Costs perf &amp; energy</td>
</tr>
</tbody>
</table>

1. Probability of fault (lower is better), and
2. Time to error from fault (larger is better)

Source: John Daly, David Mountain’s NSA Resiliency WS 02/2012
# Probability of Faults & Time to Error

<table>
<thead>
<tr>
<th>Fault</th>
<th>Probability</th>
<th>T to Error</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fans</td>
<td>High</td>
<td>Medium</td>
<td>Node down</td>
</tr>
<tr>
<td>Power Supply</td>
<td>High</td>
<td>Medium</td>
<td>Node down</td>
</tr>
<tr>
<td>CPU / SRAM</td>
<td>Very Low</td>
<td>Small</td>
<td>Node down</td>
</tr>
<tr>
<td>DRAM</td>
<td>Medium</td>
<td>Large</td>
<td>Reconfiguration</td>
</tr>
<tr>
<td>Solder Joints</td>
<td>Med-High</td>
<td>Small</td>
<td>Node down</td>
</tr>
<tr>
<td>Sockets</td>
<td>Med-High</td>
<td>Small</td>
<td>Node down</td>
</tr>
<tr>
<td>Disks</td>
<td>Mid to High</td>
<td>Large</td>
<td>Reconfiguration</td>
</tr>
<tr>
<td>NAND/PCM</td>
<td>Low-Mid</td>
<td>Large</td>
<td>Reconfiguration</td>
</tr>
<tr>
<td>Soft Errors</td>
<td>Low</td>
<td>Small</td>
<td>Clever accounting</td>
</tr>
</tbody>
</table>

Source: John Daly, David Mountain’s NSA Resiliency WS 02/2012
Deeply Scaled Technologies

**Myth:** Failure rate will increase with deep scaling

**Truth:** (near future, thermionic devices, CMOS...)
- Scaling will continue with acceptable failure rate
- But compromising performance and energy
- If the system level resiliency allows increased failures...
  - Then the technology can be aggressive
  - Benefits performance and energy

**Beyond CMOS? (far future)**
- Probabilistic?
Process Variations—Spatial, Gradual Faults

80-core research testchip

Process Variations—Spatial, Gradual Faults

Within-die and die-to-die variation impacts much higher at lower voltages

Resiliency must address spatial, gradual, and temporal faults
Soft Errors—Intermittent Faults

Beam energy spectrum compared to sea level.

NMOS diode SER dependency on voltage.

PMOS diode SER dependency on voltage.

SER dependency on NMOS diode area.

SER dependency on PMOS diode area.

Relative SER of NMOS and PMOS diodes.

Comparison of SRAM SER calculated from a calibrated 90-nm model and measured SER.

Voltage scaling of neutron SER in SRAM.

Technology scaling of neutron SER in SRAM.

Source: Several references at the end
Other Results from Literature

SER increases at NTV

Multi-bit failures become worse at NTV

SER/SRAM bit reduces with scaling

SER(bit)/cell (sea-level)

At nominal Vcc

Diffusion area reduces with scaling

1. SER/bit may reduce with scaling, but system level SER will continue to get worse
2. SER sensitivity to reduce supply voltage (NTV) needs better understanding
3. Multi-bit errors will become worse and need attention
Experiments (180, 130, 90nm)

130nm: 8490 FF * 22 dies * 10 boards = 1.87million
Recent 65 nm Experiments

Tapeout: May-2013
Debug: Aug-2013
Los Alamos: Sep-2013
OSU Nuclear-Eng: Nov-2013

<table>
<thead>
<tr>
<th></th>
<th>6T SRAM</th>
<th>8T SRAM</th>
<th>Flip-Flop</th>
</tr>
</thead>
<tbody>
<tr>
<td># of cells</td>
<td>108k</td>
<td>89k</td>
<td>17k</td>
</tr>
<tr>
<td>Cell area (µm²)</td>
<td>0.849</td>
<td>1.173</td>
<td>8.000</td>
</tr>
<tr>
<td>Block area (mm²)</td>
<td>0.306</td>
<td>0.319</td>
<td>0.182</td>
</tr>
</tbody>
</table>

R. Pawlowski et al, "Characterization of Radiation-Induced SRAM and Logic Soft Errors from 0.33V to 1.0V in 65nm CMOS", CICC, 2014

Acknowledgement: DARPA funded CREST project, Oregon State University, Prof Patrick Chiang, Robert Pawlowski, Joe Crop, and LANL (Nathan et al).
NTV exacerbates SRAM SER, multi-bit errors increase

Acknowledgement: DARPA funded CREST project, Oregon State University, Prof Patrick Chiang, Robert Pawlowski, Joe Crop and LANL (Nathan et al).
Assume: FIT Rate for SRAM & FF ~ 1e-04 with 10X uncertainty


3. Xilinx published data suggests 9e-05 (N) + 4.5e-05 (α) = 1.3e-04
Mean Time to Soft Error (Exascale)

Myth: Soft Errors are frequent
Truth: Not if they are confined
Permanent Failure Rate (VLSI Chips)

VLSI Chips are highly reliable; DRAMs more fragile
Both are VLSI, so why the difference?

V. Sridharan et. al, "A Study of DRAM Failures in the Field", SC12
Resiliency Framework Assumptions

Faults occur (relatively) infrequently, cause errors (observable)

- Diagnosis & corrective actions, do not impact performance and energy (much)

Only one fault occurs at any time in the confined area

Time to service an error or diagnose a fault is small

- Mean time to a fault is much larger than the time it takes to service a fault; assumes convergence

Fault isolation, confinement, reconfiguration, recovery and adaptation—all done in the system software (R-manager)

All levels in the stack, from Applications down to Circuits need to participate

- Error detection in hardware. Diagnosis, recovery using software
Reactive and Proactive Majors

**Reactive major**
- Detect error in hardware
- Resiliency manager (system SW) notified
- Isolate the fault (where did it happen?)
- Confine the fault (it does not impact other HW)
- Recover, reconfigure if necessary, and adapt

**Proactive major**
- Continually test the hardware (once a day, week?)
- When energy is available and not in performance critical path
- Detect marginalities, reconfigure hardware as necessary

**Hierarchical, incremental check-pointing for recovery**
- Check-pointing and recovery scheme determined by mean time to fault
Simple Detection Hardware

1. Parity/ECC covered memory with notification

<table>
<thead>
<tr>
<th>Word</th>
<th>ECC</th>
<th>ECC</th>
<th>ECC</th>
<th>ECC</th>
</tr>
</thead>
</table>

Notify where the error was detected, statistics made available

2. Parity covered datapath (not just data, but any ensemble of bits)

3. Watch-dog timers (state-machine hangs)

4. Sensors everywhere—Fans, Power supplies,...

Cost ~3% die-area & power
SW for Diagnosis and Recovery

Reactive

- Memory Errors
- Logic errors
- Watchdog Timers
- Sensors

System SW

Correct
Reconfigure

Proactive

System SW

Correct
Reconfigure

Recovery

Strategy depends on mean time to fault (T)
For large T, traditional check-pointing may be good enough
For small T, incremental, hierarchical check-pointing

# Proposed Check-pointing & Recovery

Confinement, state-store, and recovery based on:
1. Type of fault,
2. Probability of fault, and
3. Time to error

<table>
<thead>
<tr>
<th>Fault</th>
<th>Probability</th>
<th>T to Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fans</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Power Supply</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>CPU / SRAM</td>
<td>Very Low</td>
<td>Small</td>
</tr>
<tr>
<td>DRAM</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>Solder Joints</td>
<td>Med-High</td>
<td>Small</td>
</tr>
<tr>
<td>Sockets</td>
<td>Med-High</td>
<td>Small</td>
</tr>
<tr>
<td>Disks</td>
<td>Mid to High</td>
<td>Large</td>
</tr>
<tr>
<td>NAND/PCM</td>
<td>Low-Mid</td>
<td>Large</td>
</tr>
<tr>
<td>Soft Errors</td>
<td>Low</td>
<td>Small</td>
</tr>
</tbody>
</table>

Small T to error
Smaller confinement (Core level)

**Reactive measure:**
Detect in HW
Harmonize with system SW (Exec Model) to recover
Harmonizing with Execution Model

Event Driven Tasks (EDT)

1. Input Data Block
2. EDT
3. Output Data Block

- Non-preemptive completion
- Invoke dependent threads

Error? EDT

Invoke dependent threads only after error free completion

Implemented in Open Community Runtime (OCR)
# Proposed Check-pointing & Recovery

Confinement, state-store, and recovery based on:
1. Type of fault,
2. Probability of fault, and
3. Time to error

<table>
<thead>
<tr>
<th>Fault</th>
<th>Probability</th>
<th>T to Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fans</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Power Supply</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>CPU / SRAM</td>
<td>Very Low</td>
<td>Small</td>
</tr>
<tr>
<td>DRAM</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>Solder Joints</td>
<td>Med-High</td>
<td>Small</td>
</tr>
<tr>
<td>Sockets</td>
<td>Med-High</td>
<td>Small</td>
</tr>
<tr>
<td>Disks</td>
<td>Mid to High</td>
<td>Large</td>
</tr>
<tr>
<td>NAND/PCM</td>
<td>Low-Mid</td>
<td>Large</td>
</tr>
<tr>
<td>Soft Errors</td>
<td>Low</td>
<td>Small</td>
</tr>
</tbody>
</table>

Sensors detect and notify
Larger confinement (Node, socket, board)
Large T to error

**Reactive measure:**
Store EDT states for re-execution and recovery
Proposed Check-pointing & Recovery

Confinement, state-store, and recovery based on:
1. Type of fault,
2. Probability of fault, and
3. Time to error

<table>
<thead>
<tr>
<th>Fault</th>
<th>Probability</th>
<th>T to Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fans</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Power Supply</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>CPU / SRAM</td>
<td>Very Low</td>
<td>Small</td>
</tr>
<tr>
<td>DRAM</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>Solder Joints</td>
<td>Med-High</td>
<td>Small</td>
</tr>
<tr>
<td>Sockets</td>
<td>Med-High</td>
<td>Small</td>
</tr>
<tr>
<td>Disks</td>
<td>Mid to High</td>
<td>Large</td>
</tr>
<tr>
<td>NAND/PCM</td>
<td>Low-Mid</td>
<td>Large</td>
</tr>
<tr>
<td>Soft Errors</td>
<td>Low</td>
<td>Small</td>
</tr>
</tbody>
</table>

Large T to error
Smaller confinement (Node)

**Proactive measure:**
Detect marginality
Decommission node to replace component
User Experiences Reliable System

Proactive
- Test
- Reconfigure

Reactive
- Diagnosis/Correction
- Detection

Resilient system
Summary

• Understand faults
• Resiliency framework covering all types of faults
• Detection in HW, diagnosis and correction in system SW
• Then devise recovery scheme(s) considering all of the above
References

1. J. Autran, et. al., Real-time Soft-Error testing of 40nm SRAMs, *2012 IEEE International Reliability Physics Symposium (IRPS)*, Page(s): 3C.5.1 - 3C.5.9


5. P. Hazucha, et. al., Neutron Soft Error Rate Measurements in a 90-nm CMOS Process and Scaling Trends in SRAM from 0.25-pm to 90-nm Generation, IEDM 2003

6. J. Maiz, et. al., Characterization of Multi-bit Soft Error events in advanced SRAMs, IEDM 2003


10. V. Sridharan et. al, “Memory Errors in Modern Systems The Good, The Bad, and The Ugly”, ASPLOS 15