FaultSim: A Fast, Configurable Memory-Reliability Simulator for Conventional and 3D-Stacked Systems

19th Jan 2016
Prague, Czech Republic

Prashant Nair - Georgia Tech
David Roberts - AMD Research
Moinuddin Qureshi - Georgia Tech
In both computers and humans, memory is perhaps the most important attribute, and the one most likely to fail

Tse-Yu Yeh (Apple)
IMPORTANCE OF MEMORY RELIABILITY

Memory system: one of the main cause of system failure

Memory reliability seen as major challenge for Exascale

High availability servers employ Chipkill or RAID

Increasing set of challenges for future memory systems:

- Weak bits from technology scaling
- Large granularity failures as common as bit failures
- New failure modes in DRAM (e.g. TSV faults in 3D)
- New failure modes from NVRAM (disturb, endurance)

Memory reliability continues to be an important concern
Fast and accurate simulators vital to compare effectiveness of different solutions

**Goal**: Accurately evaluate memory reliability across different systems & solutions, in less than one minute
DRAM devices can encounter faults during operation. Memory reliability evaluations must account for both transient failures as well as permanent failures.
Failures occur at small and large granularities:
• Bit, Word, Column, Row, Bank, Multi-Bank

Memory reliability simulator should capture interaction of failures at different granularities
## REAL WORLD FAILURE RATE

![image]

### [SRIDHARAN+ SC’13]

<table>
<thead>
<tr>
<th>Failure Mode</th>
<th>Transient Fault Rate (FIT)</th>
<th>Permanent Fault Rate (FIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit</td>
<td>14.2</td>
<td>18.6</td>
</tr>
<tr>
<td>Word</td>
<td>1.4</td>
<td>0.3</td>
</tr>
<tr>
<td>Column</td>
<td>1.4</td>
<td>5.6</td>
</tr>
<tr>
<td>Row</td>
<td>0.2</td>
<td>8.2</td>
</tr>
<tr>
<td>Bank</td>
<td>0.8</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>18</strong></td>
<td><strong>42.7</strong></td>
</tr>
</tbody>
</table>

### Observations

1. Permanent faults >2x as likely as transient faults
2. Large granularity faults as common as bit faults

---

![image]

---

![image]
Several techniques: SECDED, Chipkill, Sparing often used in combination with periodic Scrubbing

Complex interactions of techniques with fault modes and granularities ➔ How to evaluate effectiveness?
ANALYTICAL MODELS FOR MEMORY RELIABILITY

- Complex, Cumbersome, Changes with Fault Models
- A PRDC paper* has nearly 3 page model for Chipkill

Small change in ECC → Massive changes in the model

Use empirical evaluation instead of analytical models

*Jian et. al. PRDC 2013
OVERVIEW

- WHY FAULTSIM?
- FAULTSIM: WHAT AND HOW?
- FAULTSIM: LESS THAN 1 MINUTE
- FAULTSIM: APPLIED TO 3D MEMORY
- SUMMARY
FaultSim is written in C++. Configuration at command line or file:

- **Fault Model**
  - Describes fault rate/component
  - Derived from field studies
  - Can be changed

- **Monte Carlo Simulator**
  - Describes memory system
  - Including chips per rank
  - Number of channels
  - And interconnect to processor

- **Memory Organization**
- **Interconnection Graph**
- **ECC/Repair Scheme**
- **Scrubbing Scheme**

Describes mitigation technique(s)
- Can be combination
- Used with/without scrubbing
FaultSim performs 20K*1 million interval simulations per chip for each fault type → days of simulation time
• Memory chips are organized as Fault Domains
• Fault Domain (FD) consists of Fault Ranges (FR)
• Each FR uses Address (ADDR) and Mask fields

Space Efficient Representation: Large + Small granularity faults use only one type of FR data structure
FAULT REPRESENTATION: EXAMPLE

Memory with 8 rows and 8 bits per row

• Fault ranges A, B and C (A and B intersect)
• **Mask field**: fault address bit $i$ can be 0 or 1
• **Address field**: specific address bit values where $\text{Mask}_i = 0$
• Faults intersection computed based on mask and address

<table>
<thead>
<tr>
<th>FR</th>
<th>Address</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>000001</td>
<td>011000</td>
</tr>
<tr>
<td>B</td>
<td>010000</td>
<td>000111</td>
</tr>
<tr>
<td>C</td>
<td>110000</td>
<td>000111</td>
</tr>
</tbody>
</table>
FaultSim closely follows the analytical model (within 2%)
RESULTS: SIMULATION TIME

Time for a million trials with FaultSim

<table>
<thead>
<tr>
<th>REPAIR SCHEME</th>
<th>Simulation Time (Wall Clock)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECDED</td>
<td>49.5 hours</td>
</tr>
<tr>
<td>ChipKill</td>
<td>49.2 hours</td>
</tr>
</tbody>
</table>

FaultSim still has simulation time in the order of days. How to we reduce this to less than a minute?
OVERVIEW

- WHY FAULTSIM?
- FAULTSIM: WHAT AND HOW?
- FAULTSIM: LESS THAN 1 MINUTE
- FAULTSIM: APPLIED TO 3D MEMORY
- SUMMARY
OBSERVATION: FEW FAULTS IN SYSTEM LIFETIME

FaultSim consults random number generator at-least once during each interval (20K)

System with 2 DIMMs, 9 chips each, over 7 years

<table>
<thead>
<tr>
<th>Num. Faults Encountered (Total)</th>
<th>TRIALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>92.9%</td>
</tr>
<tr>
<td>1</td>
<td>6.7%</td>
</tr>
<tr>
<td>2</td>
<td>0.2%</td>
</tr>
<tr>
<td>3+</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

Can we consult random number generator in proportion to faults, instead of every time interval?
INSIGHT: COMPUTE DISTANCE TO NEXT FAULT

The time between events in a process in which events occur continuously and independently at a constant average is \textit{exponentially distributed}.

Example: Let the likelihood of a lottery ticket be a winner be 1/1000. We buy 5000 tickets. What is the likelihood of “X” winning tickets?

Naïve Method: Draw 5000 tickets, for each ticket check if it is winner.

Distance Method: Compute distance to winning ticket using exponential distribution (avg=1000). Do until sum of distance > 5000.
Event-Based Fault Injection: When is the next fault?

Time-Stamp of all faults computed at start of simulation. Simulation skips from one fault to another.

Calls to random number reduced from 20K to 1 (or 2)
**RESULTS: SIMULATION TIME**

Time for a million trials with FaultSim

<table>
<thead>
<tr>
<th>SCHEME</th>
<th>Simulation Time (Wall Clock)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECDED (Interval Based)</td>
<td>49.5 hours</td>
</tr>
<tr>
<td>SECDED (Event Based)</td>
<td><strong>34 seconds</strong></td>
</tr>
<tr>
<td>ChipKill (Interval Based)</td>
<td>49.2 hours</td>
</tr>
<tr>
<td>Chipkill (Event Based)</td>
<td><strong>33 seconds</strong></td>
</tr>
</tbody>
</table>

FaultSim ~5000x faster with Event-Based Fault Injection ➔ reliability simulation in less than one minute
OVERVIEW

➤ WHY FAULTSIM?
➤ FAULTSIM: WHAT AND HOW?
➤ FAULTSIM: LESS THAN 1 MINUTE
➤ FAULTSIM: APPLIED TO 3D MEMORY
➤ SUMMARY
Industry moving towards 3D DRAM for higher BW
New failure modes due to Through-Silicon Via (TSV)

FaultSim can model new components like TSVs
CAPTURING THE EFFECT OF TSV FAULTS

- Data TSV Fault → Few Columns Faulty
- Address TSV Fault → 50% Memory Loss

TSVs faults manifested as column/bank failure
FaultSim used to evaluate TSV sparing in 3D memory
OVERVIEW

- WHY FAULTSIM?
- FAULTSIM: WHAT AND HOW?
- FAULTSIM: LESS THAN 1 MINUTE
- FAULTSIM: APPLIED TO 3D MEMORY
- SUMMARY
SUMMARY

• Memory-Reliability is becoming increasingly important and there is a need for evaluation tools
• We introduce FaultSim \(\rightarrow\) An efficient and fast memory reliability simulator
• FaultSim uses event-based simulation, efficient representation and quickly computable functions
• FaultSim enables evaluating memory-reliability within 2% of the analytical model
• FaultSim is \(\sim 5000\times\) faster than interval-based Monte-Carlo simulator
Clone it from github

$git clone https://github.com/Prashant-GTech/FaultSim-A-Memory-Reliability-Simulator

Running FaultSim

./faultsim --help for a list of command line parameters

./faultsim --configfile configs/DIMM_none.ini --outfile out.txt