A Generic Configurable Error Injection Agent for All On-Chip Memories

Anil Deshpande, Jaechul Park, Niharika Sachdeva, Arjun Suresh Kumar, Raviteja Gopagiri, Somasunder Kattepura Sreenath, Damandeep Saini
AGENDA

01
Introduction
- Motivation
- On-Chip Memories

02
Error Correction Code
- Description
- Existing Methodologies

03
Proposed Agent
- Solution
- Configurable parameters and patterns

04
Simulation & Results

05
Conclusions & Future Scope
Memories exist everywhere
High Probability of Bit Flips
Need for Error Correction and Detection
Agnostic verification of ECC Algorithm

Federal Voting: 18th May 2003, Belgium

Motivation
Introduction

Error Correction Code (ECC)

Proposed Agent

Simulation & Results

Conclusions & Future Scope

**Error Correction Code (ECC)**

- **Encoder**
  - Data_in[m:0]
  - Write data
  - parity_out[n:0]

- **Decoder**
  - Data_out[m:0]
  - Read data
  - Data_rd[m:0]
  - Corrected
  - Detection or uncorrectable flags

**Techniques**
- Parity Code
- Checksum
- Cyclic Redundancy Check
- Hamming code
- BCH code

**Description**
Existing Methodologies

Formal Based
- Formal Reasoning
- Formal Modelling

Simulation Based
- Error injection (Reactive)
- Memory transaction patterns required
Existing VE

Error Injection Agent

DUT IP

Virtual Interface

Error Correction Code (ECC)

Solution

Simulation & Results

Conclusions & Future Scope

1

2

DUT MEMORY

DATA IN

0 1 0 1 0 1 1 0

DATA OUT

0 1 0 1 0 1 1 0

Error Injection Agent
TB_TOP (Existing Memory Controller/SoC/StandAlone Memory)

CONFIG_ERRINJ_AGENT

Configuration Class
- Sequencer
- Driver
- Monitor
- Scoreboard
- Coverage

Virtual Interface
- Virtual Sequencer (Target VE)

DUT Memory

Introduction
Error Correction Code (ECC)
Proposed Agent
Simulation & Results
Conclusions & Future Scope

Solution
Type of Error
- SEI, DEI, TEI, MEI

Error Injection Mode
- Random, Fixed, Incremental address mode

Ratio mode
- MEI in ratios

Non stop mode
- Continuously Error injection

Fixed Error Position
- Used to create Stuck at faults

Address and Position range
- Error to be injected on a selected range of Address

Error Count
- In incremental mode, user can configure number of errors injected.

Number of ECC’s
- Parallel testing for those blocks whose data width is covered by different ECC modules

Use Reference Memory
- To enable or disable checks for Memory DATA IN and DATA OUT

March C
{↑ w0, ↑ (r0, w1), ↑ (r1, w0), ↓ (r0, w1), ↓ (r1, w0), ↓ r0}

Scan
{↑ w0, ↑ r0, ↑ w1, ↑ r1}

Checkerboard
{↑ w0/1, ↑ r0/1; ↑ w1/0, ↑ r1/0}

Random
{↑ w, ↑ r, ↓ w, ↓ r, ↓ w, ↓ r}
**SRAM Memory Signals:**
Indicates Read Operation on XA address

**EccErrInject:**
Tells if Error was injected

**ErrinjPos:**
Tells which bit position was flipped

---

**Simulation & Results**

![Simulation waveform](image-url)
<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Data width</th>
<th>No of ECC Blocks</th>
<th>Number of coverage bins (RTL design code)</th>
<th>Simulation time saved to achieve 100% coverage</th>
<th>Test Bench development time saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRAM</td>
<td>64 data bits + parity bits</td>
<td>1</td>
<td>7100</td>
<td>25%</td>
<td>40%</td>
</tr>
<tr>
<td>MRAM</td>
<td>2*(128 data bits + parity bits)</td>
<td>2</td>
<td>18400*2</td>
<td>35%</td>
<td>45%</td>
</tr>
<tr>
<td>SRAM</td>
<td>32 bits + parity bits</td>
<td>1</td>
<td>400</td>
<td>20%</td>
<td>50%</td>
</tr>
<tr>
<td>FLASH</td>
<td>32 bits + parity bits</td>
<td>1</td>
<td>1589</td>
<td>25%</td>
<td>35%</td>
</tr>
</tbody>
</table>

On an average 0.5x time saved for a single instance of Memory with ECC
Early detection of bugs in embedded ECC logic

Comprehensive and early coverage closure for ECC circuit logic

No overhead in simulation time

Easy to integrate with any UVM environment

Optimizing on memory test patterns to close verification faster

Extension on actual memory netlists (non-behavioral model) and then compare the performance metrics.
Thanks !!!