MCXplore: An Automated Framework for Validating Memory Controller Designs

https://git.uwaterloo.ca/caesr-pub/mcxplore

Mohamed Hassan and Hiren Patel
{mohamed.hassan, hiren.patel}@uwaterloo.ca

University of Waterloo
Background

- There has been a focus on validating processing elements
- Main memory is becoming a vital component in almost all computing systems
Background

$t_{RC}$

$t_{RCD}$  $t_{CCD}$  $t_{RTP}$  $t_{RP}$

req1  req2  req3  req4

Memory Controller

Requestors

Logical Address

Address Mapping

Physical Address

Command Generation

Command Queues

Command Arboration

Interface Queues

Banks

Columns

Rows

Row-Buffer

26-Mar-16

Mohamed Hassan/ University of Waterloo
Simulation-based Validation

**Benchmarks**

- + Time and effort conserving
- - May not be memory intensive
- - Lack easy-to-analyse memory patterns
- - Do not explore the state space of the memory subsystem properties

**Exhaustive Tests**

- + Guaranteed coverage
- - Very time and resource consuming (may not be possible)

**Manual Tests**

- + Allows for directed testing to cover specific properties
- - Time consuming
- - Prone to human errors

**Random Tests**

- + Moderate time and effort
- - Questionable test coverage
Outline

• MCxplore, objective and flow

• Validation example from the MC frontend

• Validation example from the MC backend

• Additional features

• Conclusions
Contributions

**Formal models**

- **Request Model:** The interrelation amongst memory requests
- **Command Model:** Interactions between memory commands

**Model Checking**

- **Test Plan:** Precisely specify test properties in TLA
- **Separate test template from actual tests**
- **Optimal Tests:** With minimum number of requests

**Validation Methodology**

- **Golden metric to validate results**
- **Automated test generation**

**MCXplore**

**Extensive Testing**

- **Vital sequence patterns and test plans for state-of-the-art MCs**
- **Test suites for commodity MC policies**

**Contributions**

- **Extensive Testing**
- **Validation Methodology**
- **MCXplore**
- **Formal models**
- **Model Checking**
Proposed Process

**Step 1:** Test Plan
- Test Plan

**Step 2:** Test Template Generation
- New SPECs
  - Yes
    - New
  - No
    - Pre-defined SPECs
    - Select
      - Command model
      - Request model
      - Model Checker

**Step 3:**
- Counter-example exist?
  - Yes
    - Test Template
  - No

**Step 4:** Test Suite Generation
- #Tests
  - Parser
    - Test suite
    - Addr. map.
    - Syntax

**Step 5:** Diagnosis and Report
- Expected behaviour
  - MC
  - Compare
  - Equal?
    - Yes
      - Report Correctness
    - No
      - Report Bug
        - More Diagnosis?
          - Yes
          - More Diagnosis?
          - No

- END
Request Model

32b addr.

\[ 2^{32} \times n \times 2 \] possibleities

\[ \text{either same or diff.} \]

\[ 2^5 \times n \times 2 \] possibleities

5 segments

26-Mar-16

Mohamed Hassan/ University of Waterloo
Command Model

- Simplifies testing properties that are related to timing constraints and command generation

- Enables specifying the timing constraints and MCXplore automatically generates the test sequence that exercises these constraints
Evaluation

- DRAMsim2 as the base MC
- Insert bugs
- Apply proposed methodology
Example: XOR Address Mapping

<table>
<thead>
<tr>
<th>LS row bits</th>
<th>Bank bits</th>
<th>New bank bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>001</td>
<td>000</td>
</tr>
<tr>
<td>010</td>
<td>001</td>
<td>011</td>
</tr>
</tbody>
</table>
XOR Address Mapping

Test Plan

1. Optimal memory pattern for the XOR mapping
2. A stream of read accesses where we change the bank interleaving ratio per test, requests targeting the same bank are accessing different rows
XOR Address Mapping

(2) Specs.

$$\text{LTLSPEC}$$

$$G((t_{req} = 6 \land t_{hit} = 0 \land t_{intr} = 0) \rightarrow \neg F(t_{req} = 10 \land t_{hit} = 4 \land t_{intr} = 4))$$
• Model checker produces a counter-example for each specification.
• Each template has a bank interleaving percentage between 0% and 100%
## XOR Address Mapping

<table>
<thead>
<tr>
<th>Address</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>R</td>
<td>00000 000</td>
</tr>
<tr>
<td>0x00080000</td>
<td>R</td>
<td>00001 000</td>
</tr>
<tr>
<td>0x00100000</td>
<td>R</td>
<td>00010 000</td>
</tr>
<tr>
<td>0x00180000</td>
<td>R</td>
<td>00011 000</td>
</tr>
<tr>
<td>0x00200000</td>
<td>R</td>
<td>00100 000</td>
</tr>
<tr>
<td>0x00280000</td>
<td>R</td>
<td>00101 000</td>
</tr>
<tr>
<td>0x00290000</td>
<td>R</td>
<td>00101 001</td>
</tr>
<tr>
<td>0x002a0000</td>
<td>R</td>
<td>00101 010</td>
</tr>
<tr>
<td>0x002b0000</td>
<td>R</td>
<td>00101 011</td>
</tr>
<tr>
<td>0x002c0040</td>
<td>R</td>
<td>00101 100</td>
</tr>
</tbody>
</table>

- **same bank different rows**
- **different rows same bank 40% inter.**
XOR Address Mapping

- Elect memory utilization as a golden metric
  - It does not require any special debugging capabilities inside the MC

\[ Uti_{xor} = \frac{4tBUS}{tFAW} \]
Diagnosis

\[ \ldots r_2 r_1 r_0 \quad b_2 b_1 b_0 \]

- **Bug**
  \[ b_2 b'_1 b'_0 \]

- **Correct XOR**
  \[ b'_2 b'_1 b'_0 \]

- **No XOR**
  \[ b_2 b_1 b_0 \]
Timing Parameters Validation

• Each test is designed to maximize the impact of the timing parameter under test while eliminating or minimizing the effect of all other parameters

• Timing parameters dependency graph
• Each test is designed to maximize the impact of the timing parameter under test while eliminating or minimizing the effect of all other parameters

• Timing parameters dependency graph
Example: Read-to-Precharge Constraint

Test Plan

1. Target: validate $t_{RTP}$
2. A valid command sequence is A followed by one or more R then a P to close the row followed by an A to a different row
Read-to-Precharge Constraint

\[ LTLSPEC \ G! (\text{num}_{tRTP} \geq 1) \]
\[ RtoP = tRTP + tBUS - tCCD \]
Read-to-Precharge Constraint

Test Plan

<table>
<thead>
<tr>
<th>Address</th>
<th>Action</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>0x00000040</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>0x00000080</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>0x000000c0</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>0x001000c0</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

last request if for a different row \( \Rightarrow \) issue P
Example: Read-to-Precharge Constraint

Golden result

\[
U_{ti_{RTP}} = \frac{4tBUS}{tRCD + 3tCCD + tRTP + tRP}
\]
Read-to-Precharge Constraint

(6)
Diagnosis

BW Utilization vs. tRTP (Cycles)

- Standard-violation
- Non-optimal
### Configurability

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Address mapping</th>
<th>Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Addr. length</td>
<td>• Row mask</td>
<td>• Transaction:</td>
</tr>
<tr>
<td>• Output syntax</td>
<td>• Column mask</td>
<td>• rd, wr, random, sw%</td>
</tr>
<tr>
<td>• Transaction size</td>
<td>• Rank mask</td>
<td>• Row:</td>
</tr>
<tr>
<td>• Number of requests</td>
<td>• Bank mask</td>
<td>• hit, conflict, random, linear, locality %, custom</td>
</tr>
<tr>
<td></td>
<td>• Channel mask</td>
<td>• Rank/ Bank/ Channel:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Same, linear, random, linear,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>interleave%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• .....</td>
</tr>
</tbody>
</table>
## Test Suites

<table>
<thead>
<tr>
<th>Suite</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegressionSuite</td>
<td>includes tests that cover all combinations of the configuration parameters</td>
</tr>
<tr>
<td>PoliciesSuite</td>
<td>includes tests that test most commonly used policies of commodity memory controllers such as page policies, address mapping and arbitration schemes</td>
</tr>
<tr>
<td>TimingSuite</td>
<td>includes tests to detect any timing violations in most timing constraints</td>
</tr>
</tbody>
</table>
MCXplore is design-independent
  - Two formal models at different granularities to capture MC behaviors
  - A precise methodology to define test plans
- Validated state-of-the-art commercial MC policies
- Highlight interesting test patterns and use memory utilization as a golden metric
- Three test suites to validate and evaluate any new MC feature
- It is open-source!
  - [https://git.uwaterloo.ca/caesr-pub/mcxplore](https://git.uwaterloo.ca/caesr-pub/mcxplore)