



DESIGN, AUTOMATION & TEST IN EUROPE

14 – 18 March, 2016 · ICC · Dresden · Germany

The European Event for Electronic
System Design & Test

***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

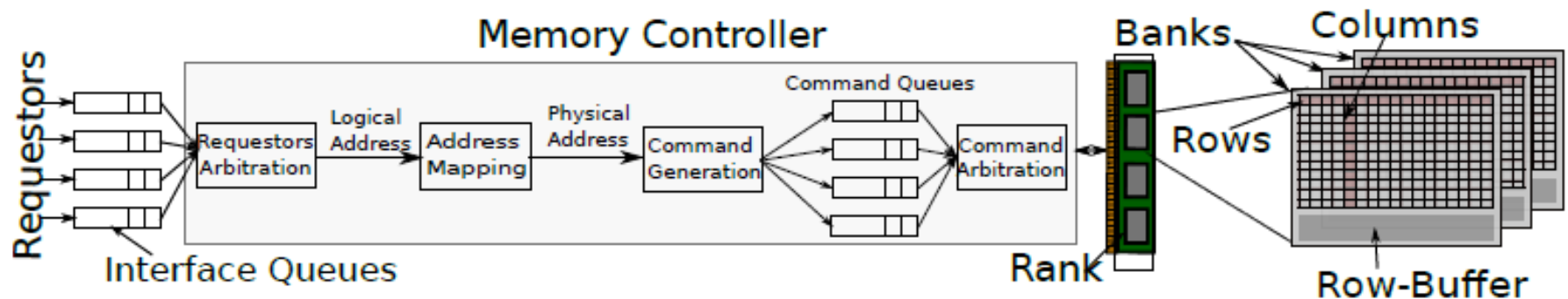
University of Waterloo



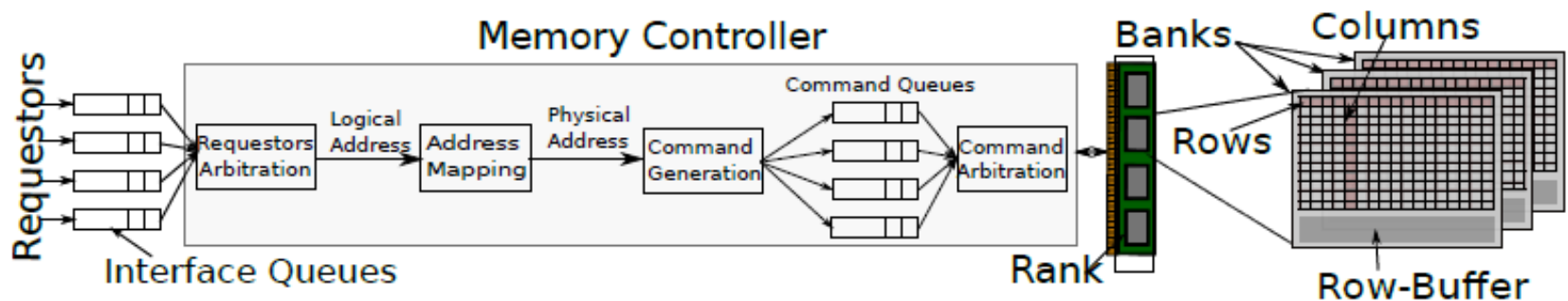
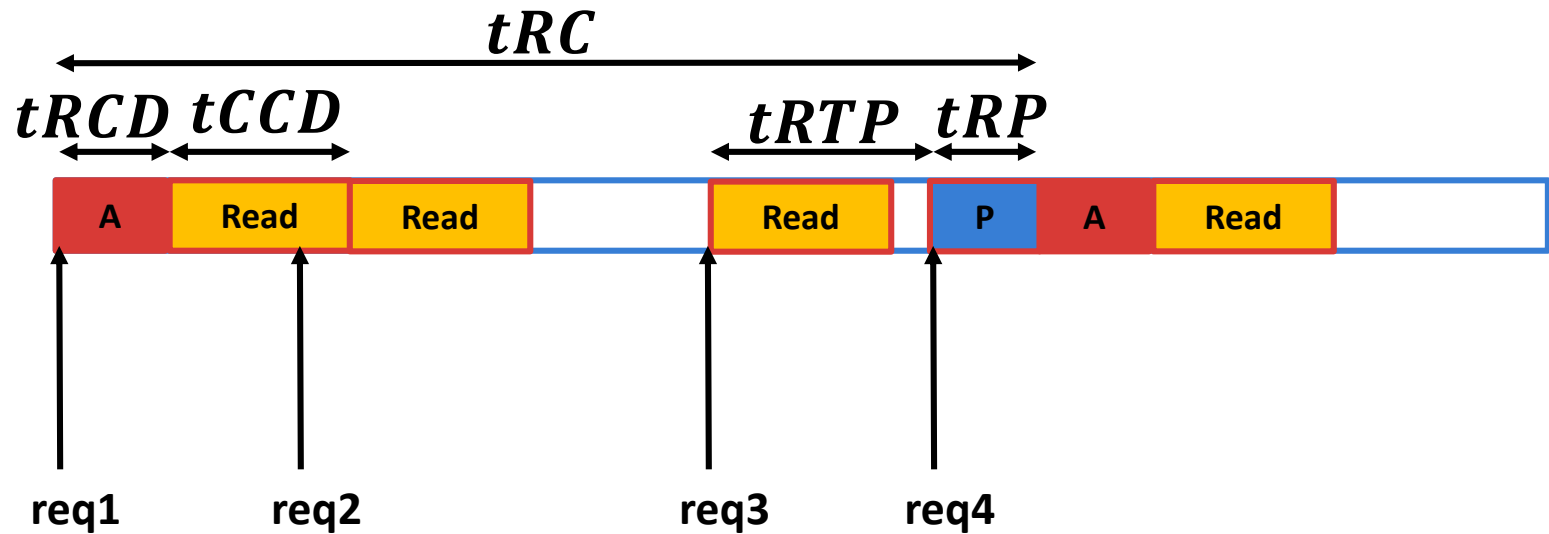
UNIVERSITY OF
WATERLOO

Background

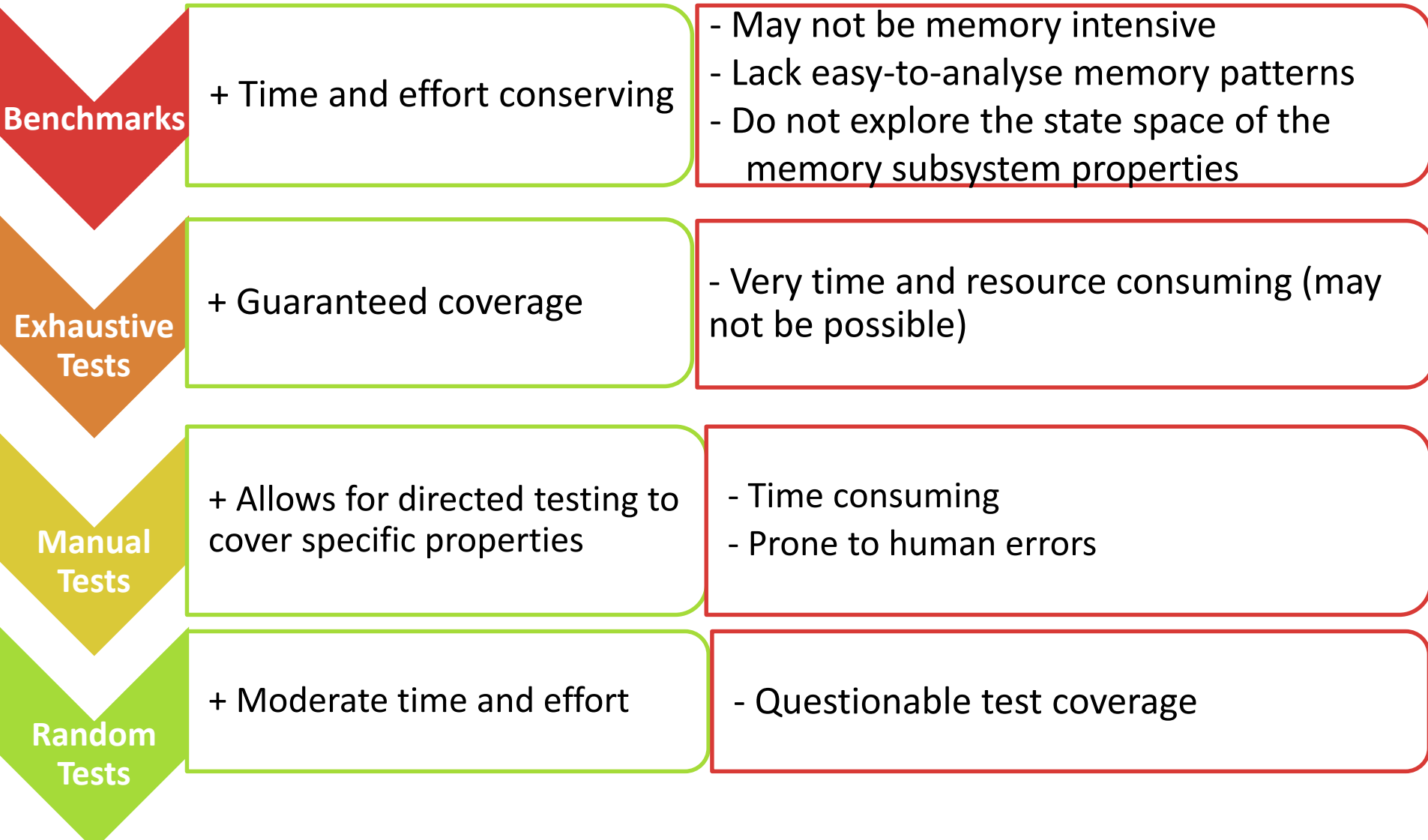
- There has been a focus on validating processing elements
- Main memory is becoming a vital component in almost all computing systems



Background



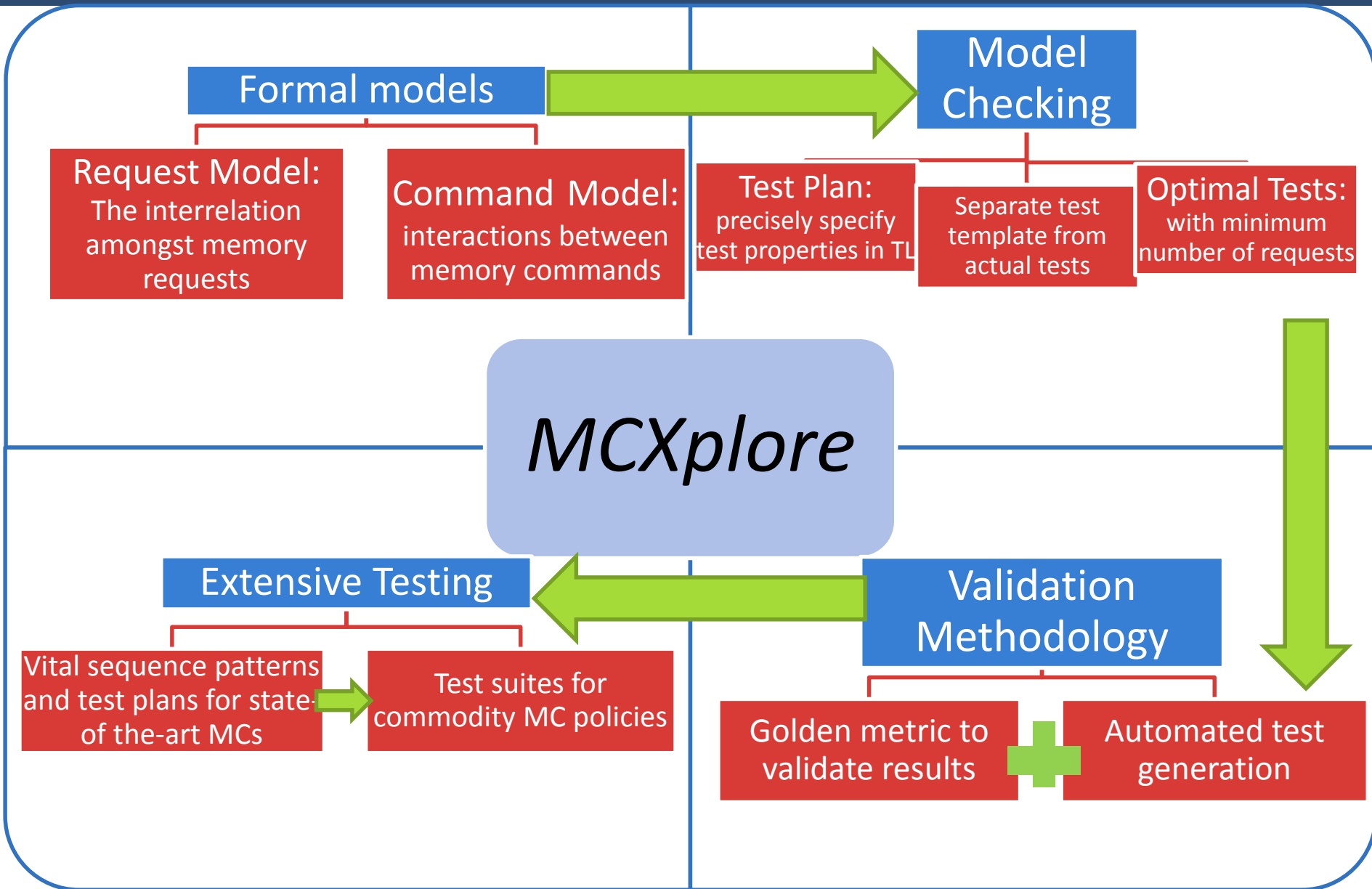
Simulation-based Validation



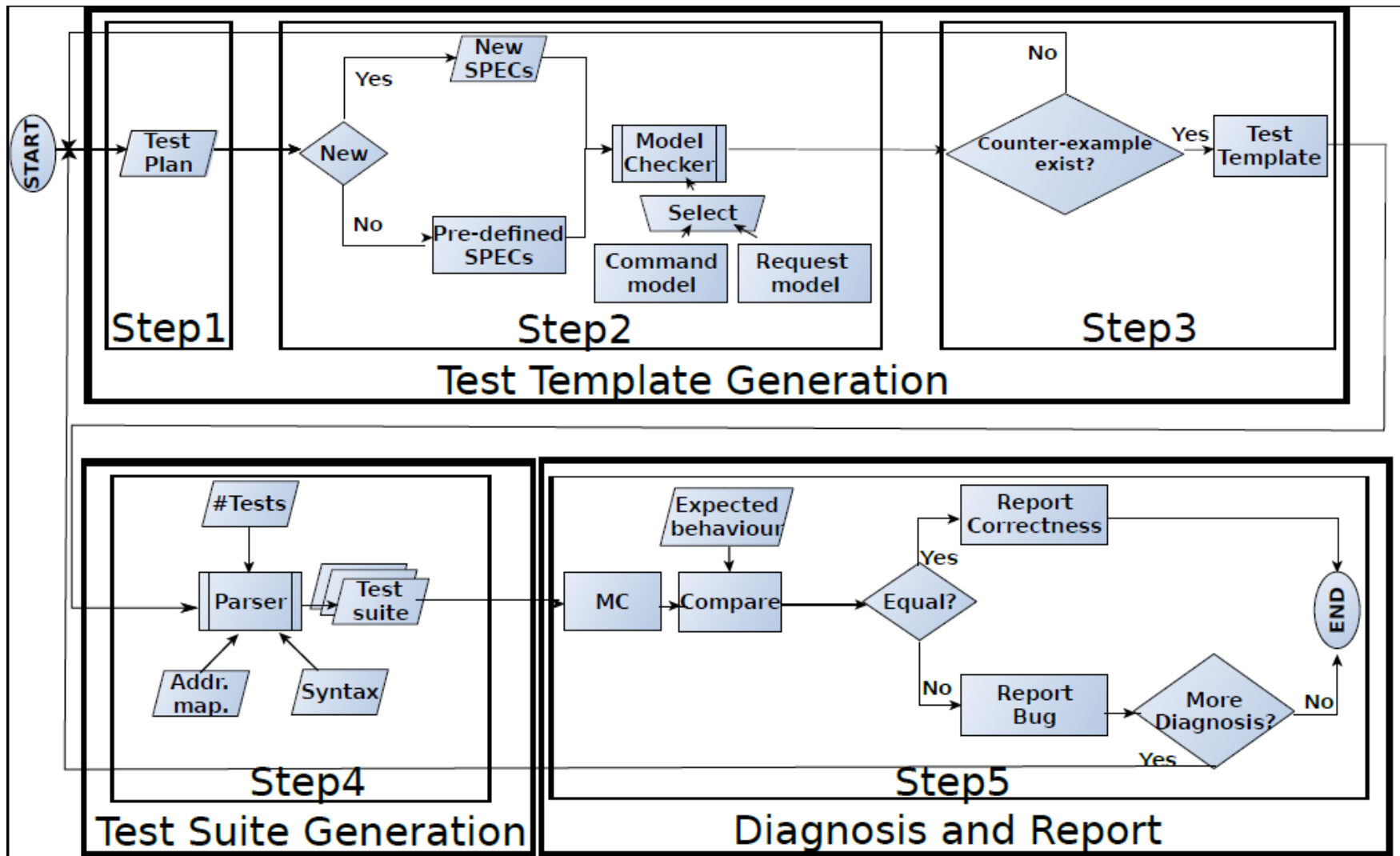
Outline

- **MCxplore, objective and flow**
- **Validation example from the MC frontend**
- **Validation example from the MC backend**
- **Additional features**
- **Conclusions**

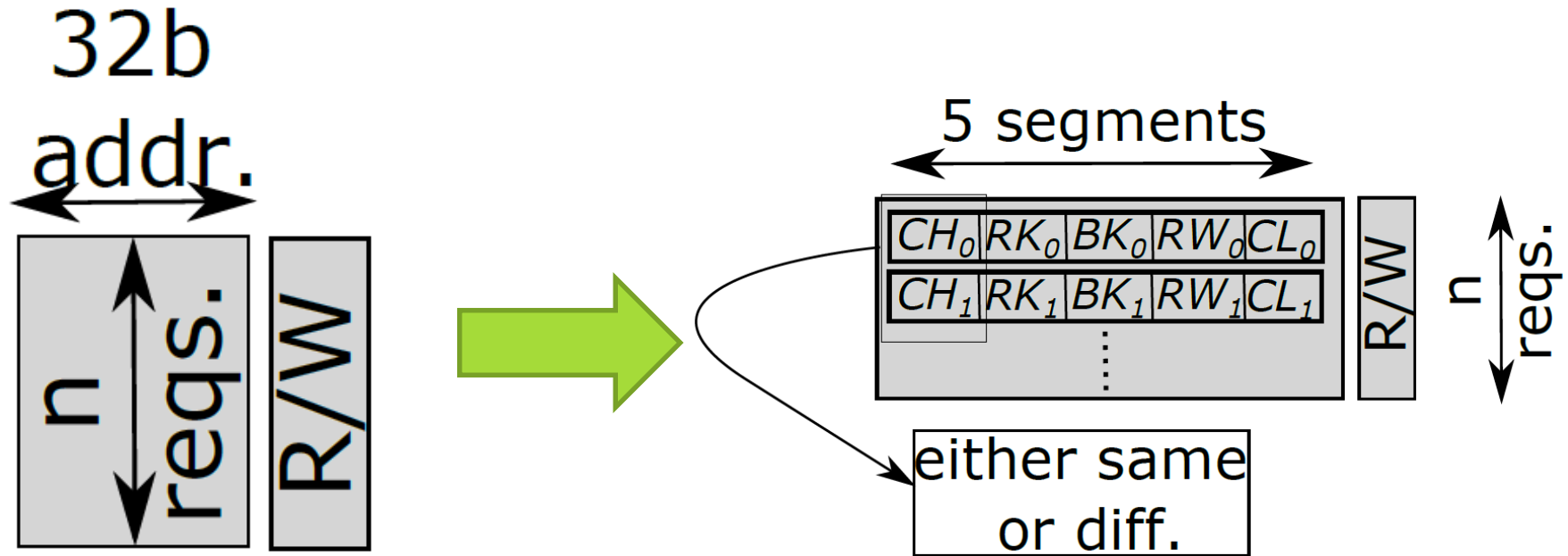
Contributions



Proposed Process



Request Model

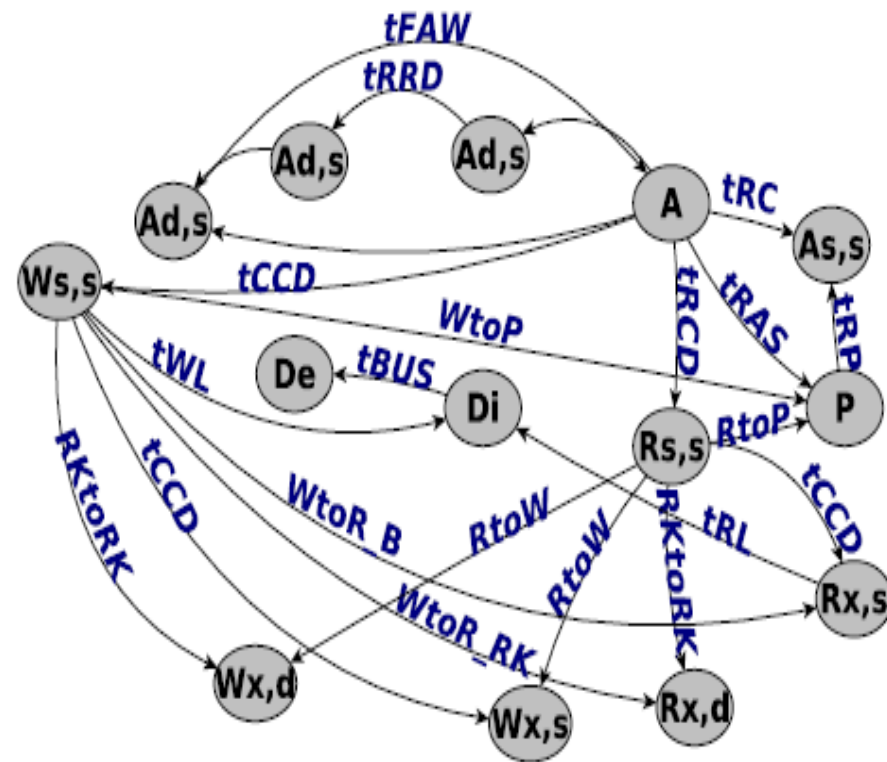


$2^{32} \times n \times 2$
possibilities

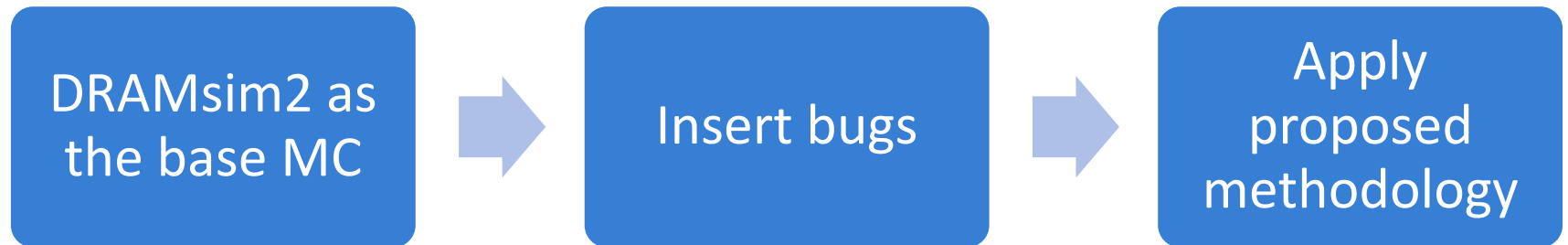
$2^5 \times n \times 2$
possibilities

Command Model

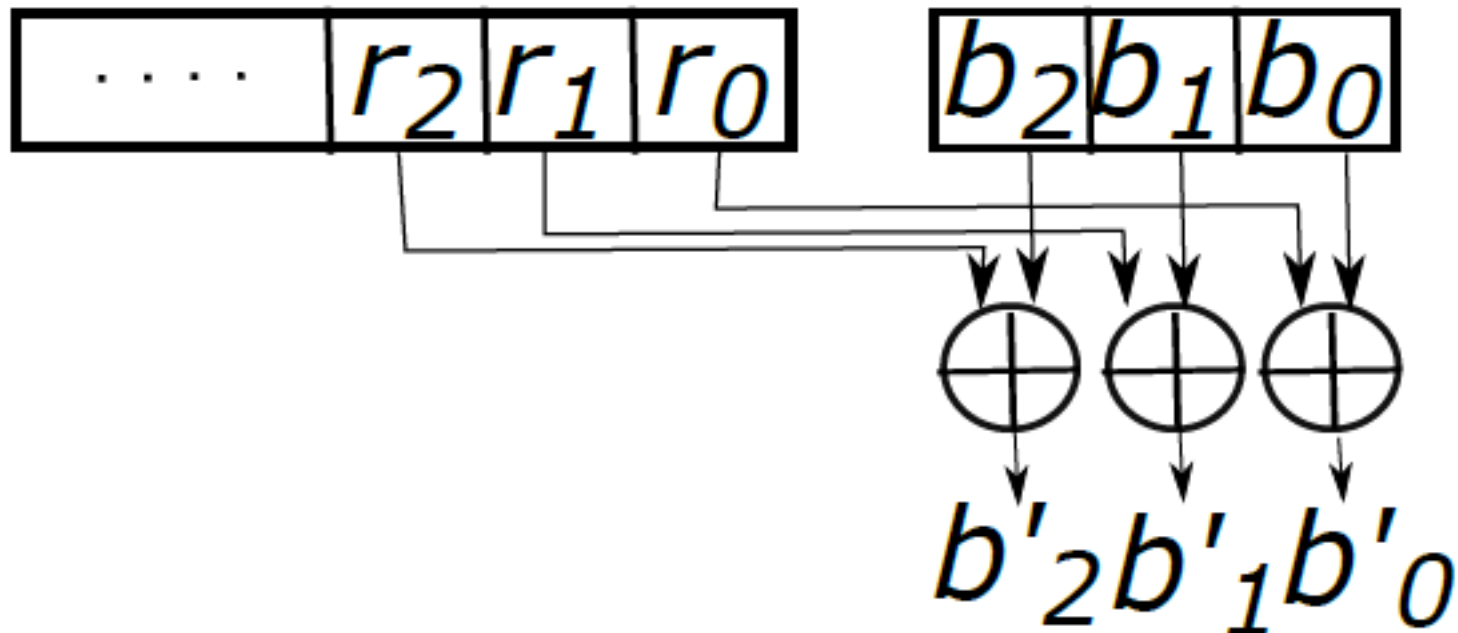
- **Simplifies testing properties that are related to timing constraints and command generation**
- **Enables specifying the timing constraints and *MCXplore* automatically generates the test sequence that exercises these constraints**



Evaluation



Example: XOR Address Mapping



LS row bits	Bank bits	New bank bits
001	001	000
010	001	011

XOR Address Mapping

(1) Test Plan

- Optimal memory pattern for the XOR mapping
- 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.

LTLSPEC

$G((t_{req} = 6 \wedge t_{hit} = 0 \wedge t_{intr} = 0) \rightarrow$
 $\neg F(t_{req} = 10 \wedge t_{hit} = 4 \wedge t_{intr} = 4))$

XOR Address Mapping

(3) Test Template

- Model checker produces a counter-example for each specification.
- Each template has a bank interleaving percentage between 0% and 100%

XOR Address Mapping

(4) Test Suite

0x00000000	R
0x00080000	R
0x00100000	R
0x00180000	R
0x00200000	R
0x00280000	R
0x00290000	R
0x002a0000	R
0x002b0000	R
0x002c0040	R

00000	000
00001	000
00010	000
00011	000
00100	000
00101	000
00101	001
00101	010
00101	011
00101	100

same bank
different rows

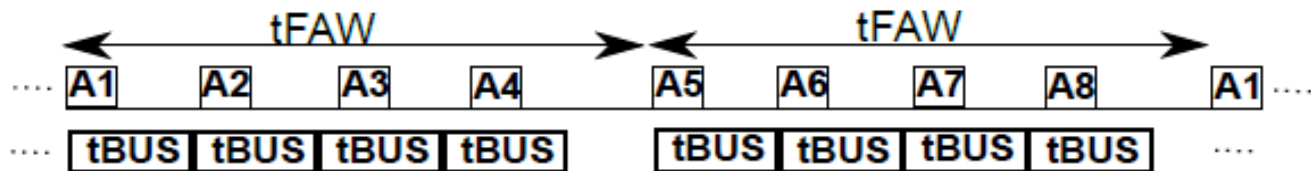
different rows
same bank
40% inter.

XOR Address Mapping

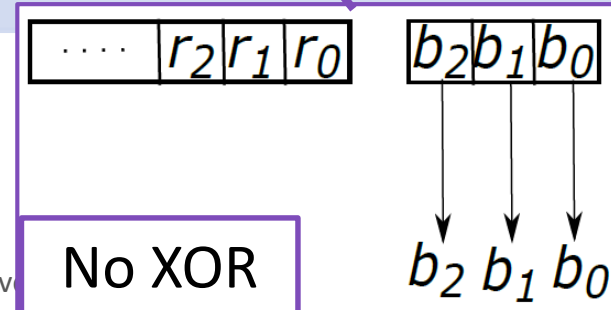
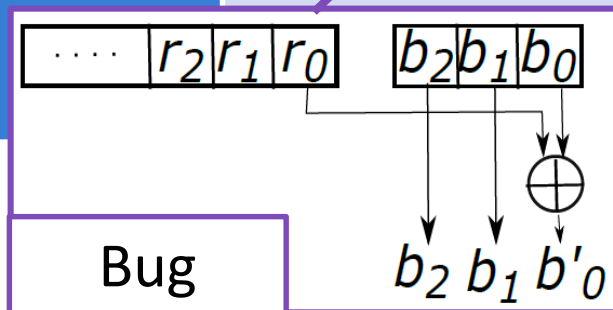
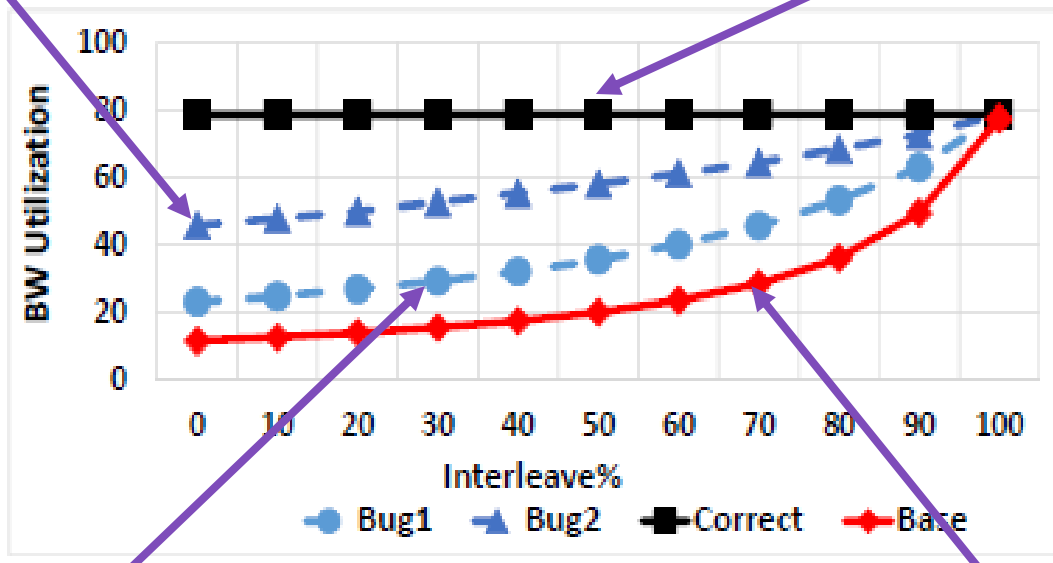
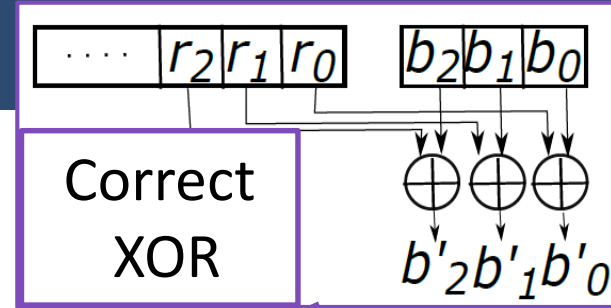
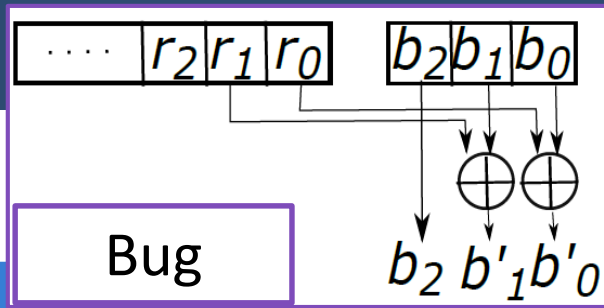
- Elect memory utilization as a golden metric
 - It does not require any special debugging capabilities inside the MC

(5)
Golden
metric

$$Uti_{xor} = \frac{4t_{BUS}}{t_{FAW}}$$

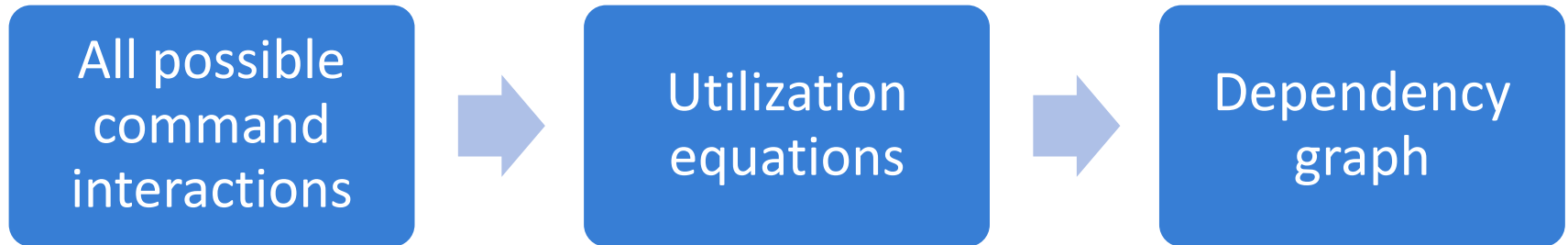


(6)
Diagnosis



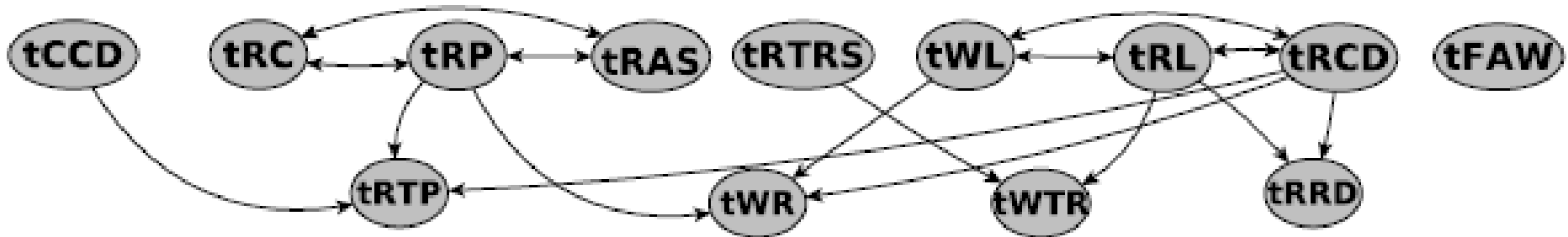
Timing Parameters Validation

- 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



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



Example: Read-to-Precharge Constraint

(1) Test Plan

- Target: validate $tRTP$
- 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

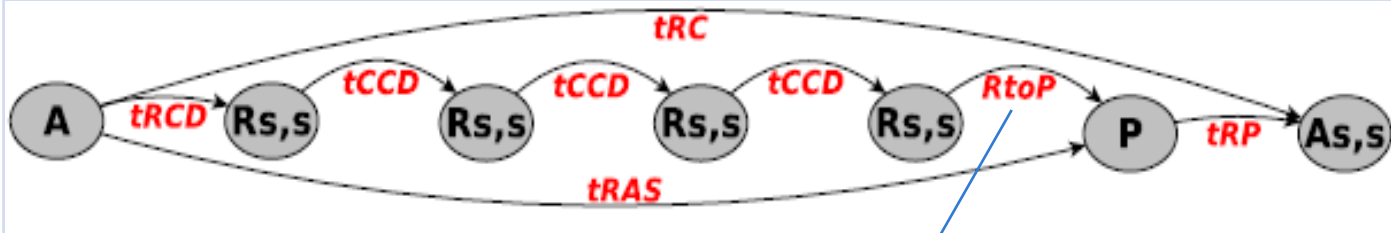
(2)
Specs.

LTLSPEC

$G! (\text{num}_{\text{tRTP}} \geq 1)$

Read-to-Precharge Constraint

(3)
Test
Template



$$RtoP = tRTP + tBUS - tCCD$$

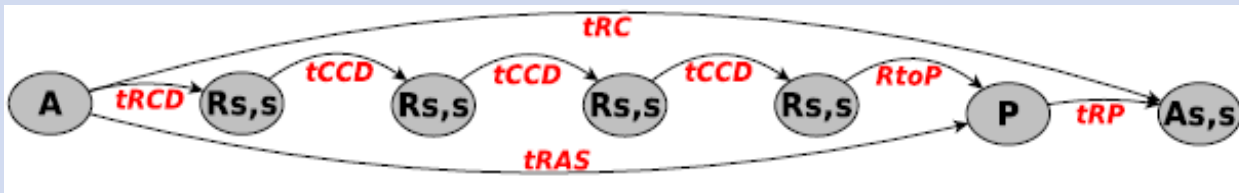
Read-to-Precharge Constraint

(1) Test Plan

0x00000000	R
0x00000040	R
0x00000080	R
0x000000c0	R
0x001000c0	R

00000	000
00000	000
00000	000
00000	000
00001	000

last request if
for a different
row → issue P

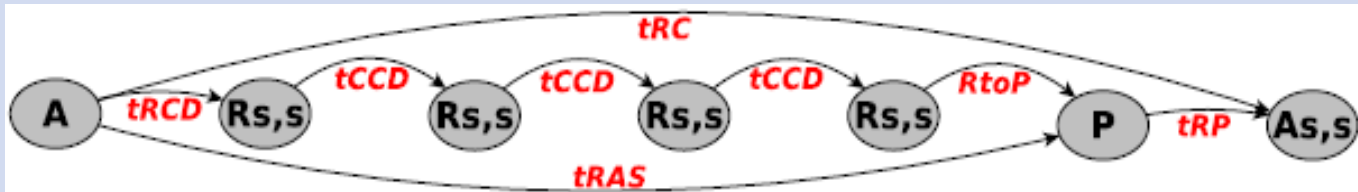


Example: Read-to-Precharge Constraint

(5)

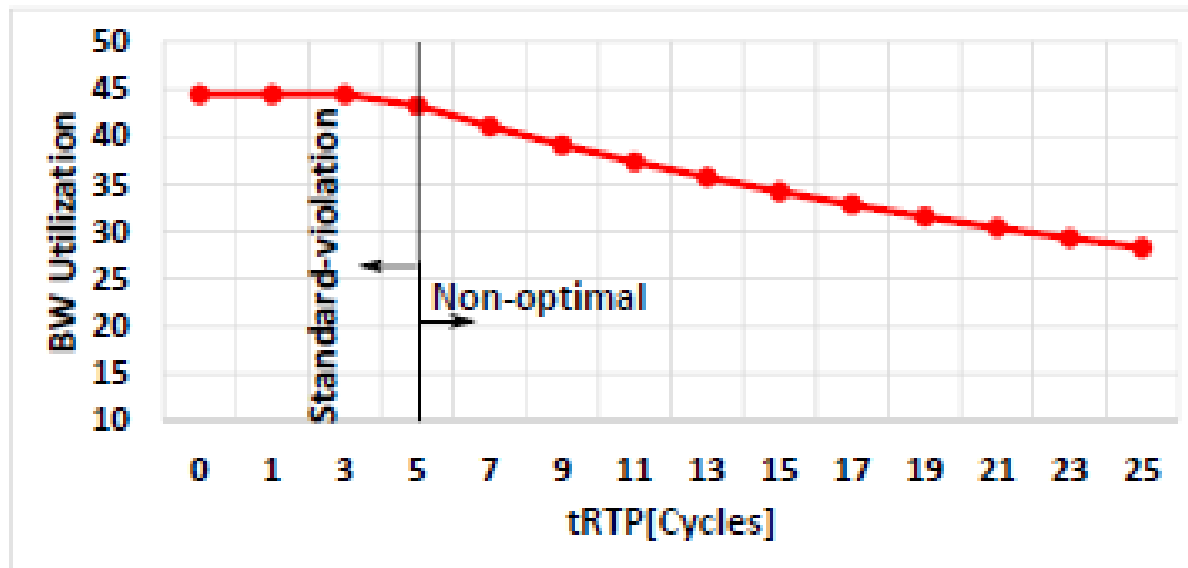
Golden
result

$$Uti_{RTP} = \frac{4t_{BUS}}{t_{RCD} + 3t_{CCD} + t_{RTP} + t_{RP}}$$



Read-to-Precharge Constraint

(6) Diagnosis



Configurability

Syntax

- Addr. length
- Output syntax
- Transaction size
- Number of requests

Address mapping

- Row mask
- Column mask
- Rank mask
- Bank mask
- Channel mask

Patterns

- Transaction:
 - rd, wr, random, sw%
- Row:
 - hit, conflict, random, linear, locality %, custom
- Rank/ Bank/ Channel:
 - Same, linear, random, interleave%
-

Test Suites

Suite	Description
RegressionSuite	includes tests that cover all combinations of the configuration parameters
PoliciesSuite	includes tests that test most commonly used policies of commodity memory controllers such as page policies, address mapping and arbitration schemes
TimingSuite	includes tests to detect any timing violations in most timing constraints

Conclusion

- ***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>