Universität Bremen

German Research Center for Artificial Intelligence

Automated Equivalence Checking Method for Majority based In-Memory Computing on ReRAM Crossbars

Arighna Deb[1], Kamalika Datta[2], Muhammad Hassan[2,4], Saeideh Shirinzadeh[2,3], Rolf Drechsler[2,4]

[1]KIIT University, Bhubaneswar, India
[2]DFKI GmbH, Bremen, Germany
[3]Fraunhofer ISI, Karlsruhe, Germany
[4]University of Bremen, Bremen, Germany

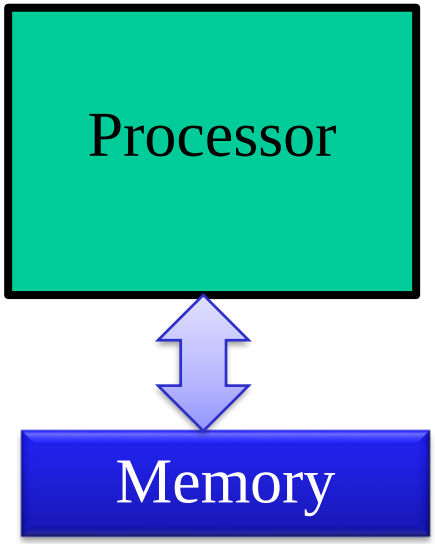
muhammad.hassan@dfki.de

Bundesministerium für Bildung und Forschung
16ME0117
AUTOASSERT

Outline

- ReRAM device
- Motivation
- Proposed verification flow
- Results
- Conclusion

Memory Wall Problem



Processor-memory speed gap

Major performance bottleneck

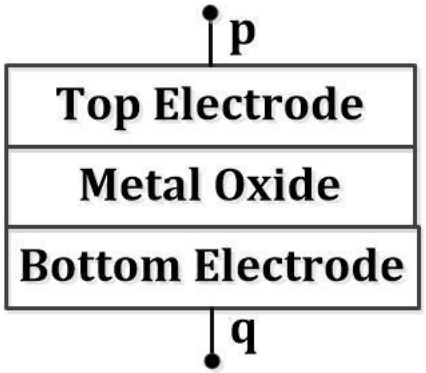
Possible Solution

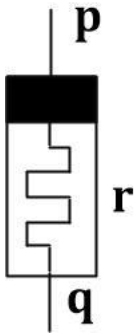
- Reduce the data transfer

Computing within memory (In-Memory Computing)

ReRAM

- Resistive Random Access Memory
- Low resistance state – logic 1
- High resistance state – logic 0





p	q	r	r_n
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	0

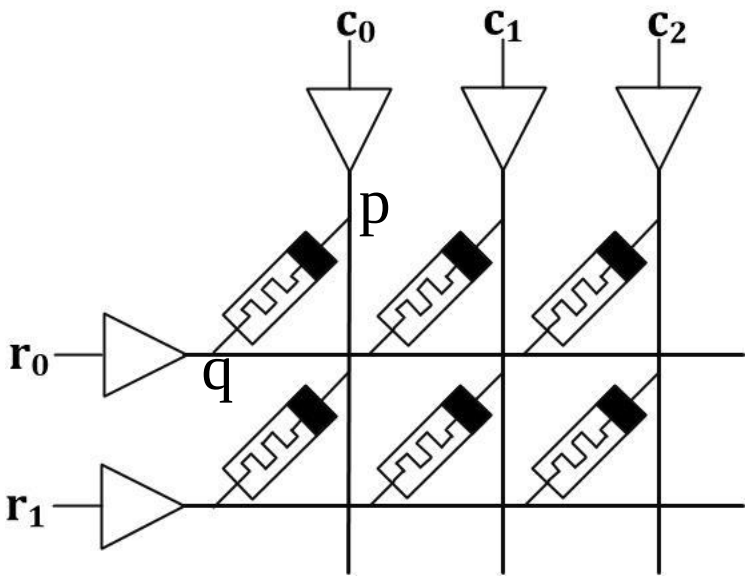
p	q	r	r_n
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	1

$$f(p,q,r) = p\bar{q} + pr + \bar{q}r$$

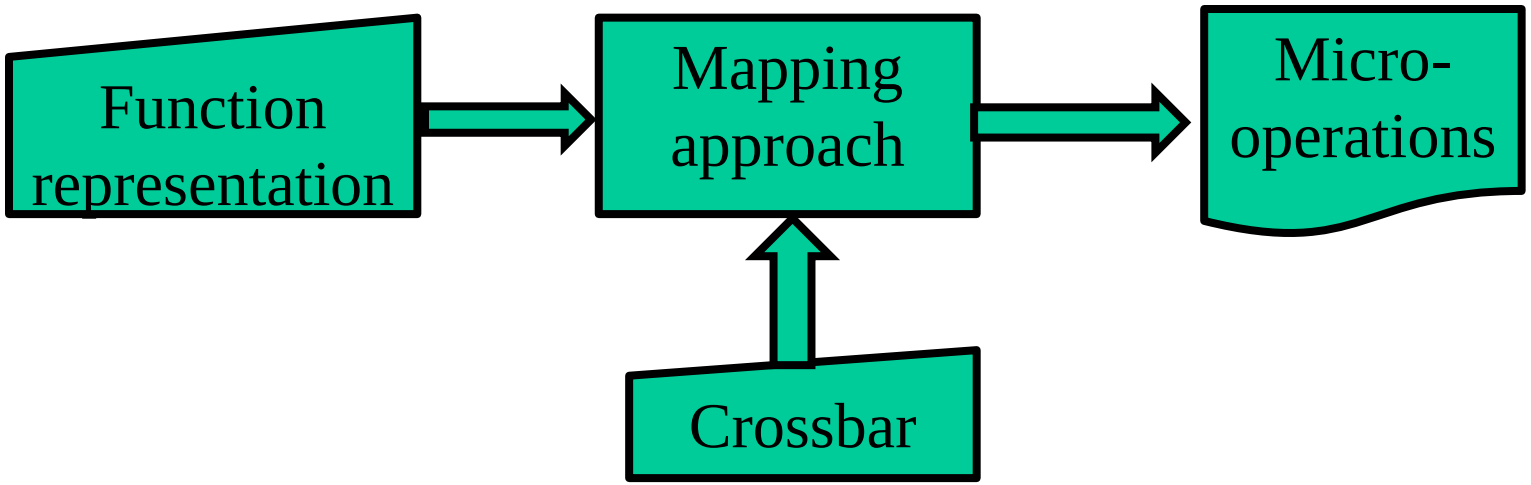
ReRAM Crossbar

bitlines

wordlines

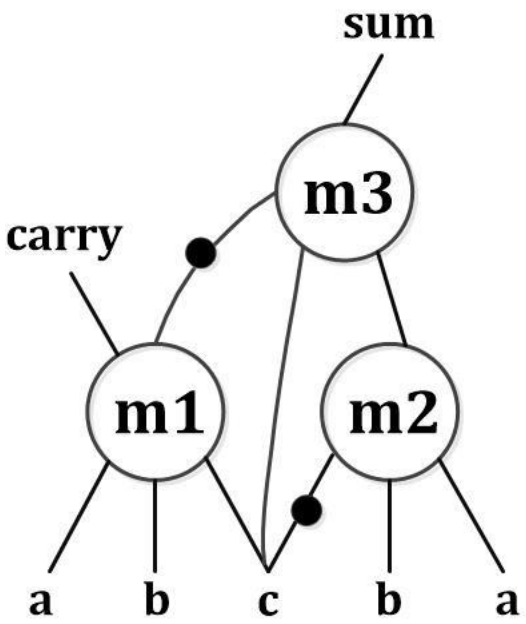


Logic Synthesis

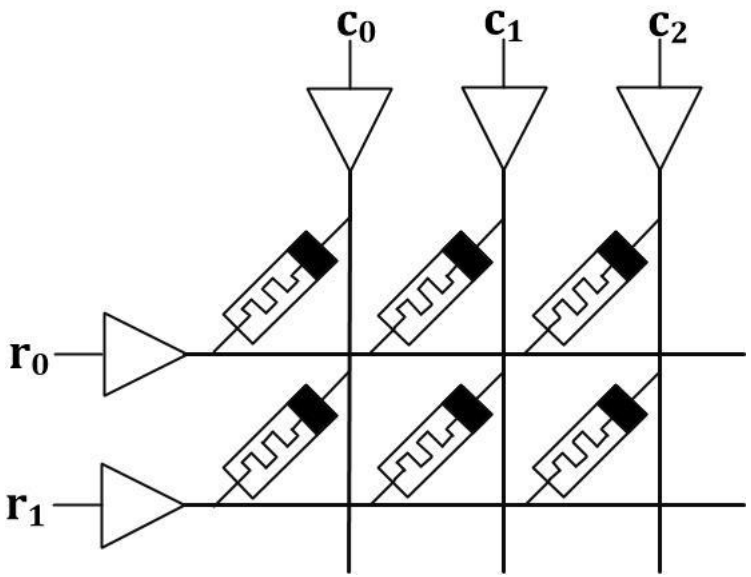


Example

- Majority function $f(a,b,c) = ab + bc + ac$



Mapping



Majority Inverter Graph (MIG)

ReRAM Crossbar

Micro-operation File Format

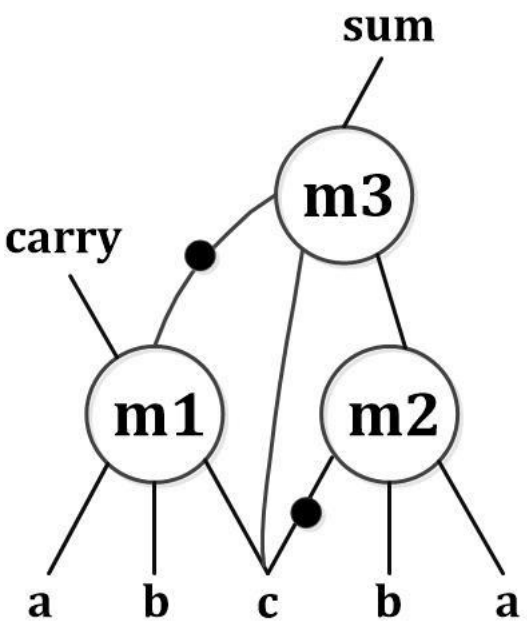
Input-output of crossbar

Step	ReRAM	p	q	r	r+
1	0x0	a	0	?	a
	0x1	b	0	?	b
	0x2	c	0	?	c
2	1x0	0	1	?	0
	1x1	0	1	?	0
	1x2	0	1	?	0
3	1x0	1	b	0	$\bar{b}$
4	1x2	c	0	0	c
5	1x2	a	$\bar{b}$	c	m1
6	1x1	b	0	0	b
7	1x1	a	c	b	m2
8	1x1	c	m1	m2	m3

Resulting representation

```
.input a b c
.output carry sum
 0 0 ¥a 1 ¥b 2 ¥c
 1 True 0 False 1 False 2 False
 1 0x1 0 True
 1 False 2 0x2
 1 1x0 2 0x0
 1 False 1 0x1
 1 0x2 1 0x0
 1 1x2 1 0x2
 ¥sum 1x1
 ¥carry 1x2
```

Research Question



Original function

Functionally Equivalent?

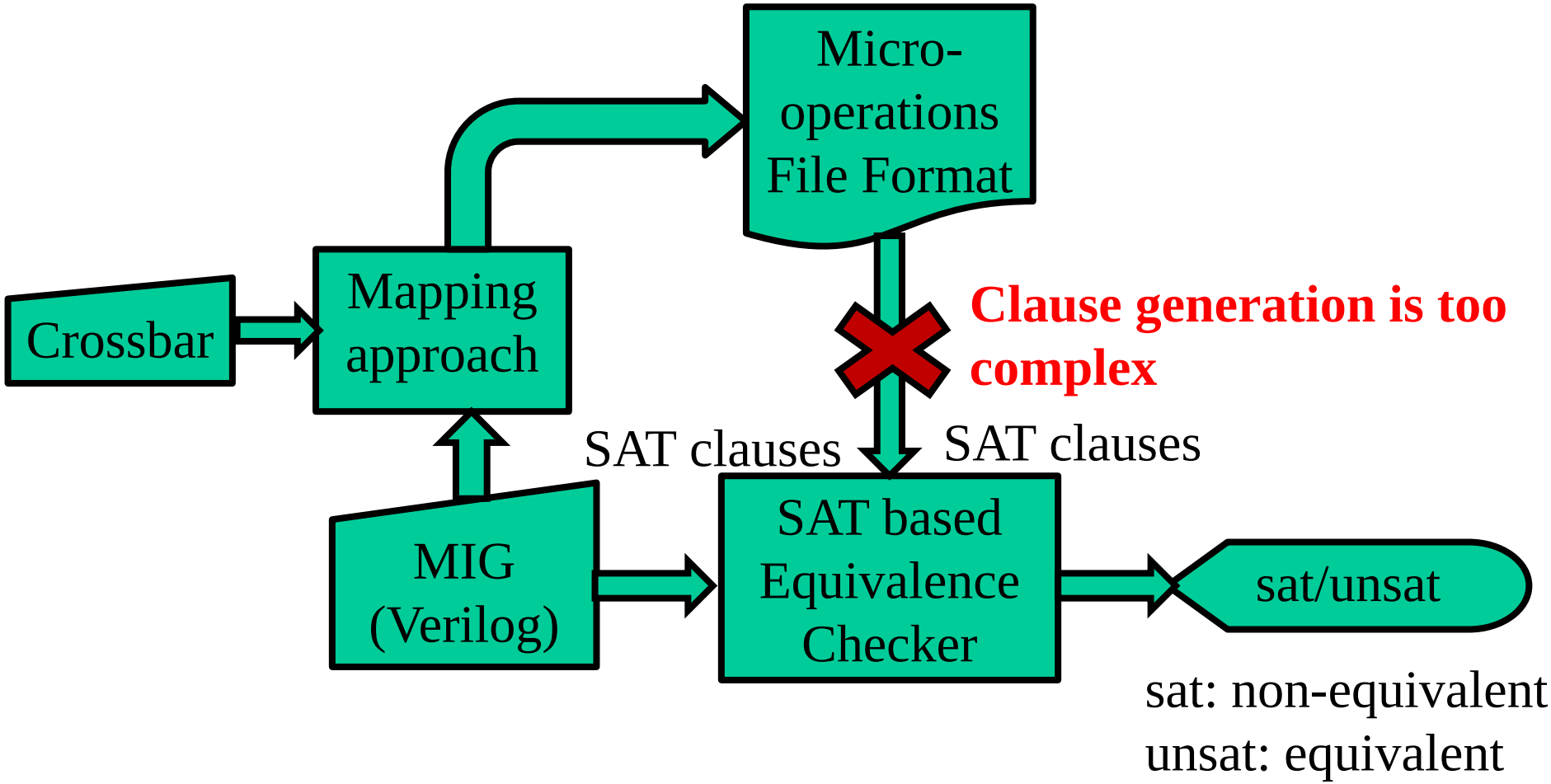
```
.input a b c
.output carry sum
0 0 ¥a 1 ¥b 2 ¥c
1 True 0 False 1 False 2 False
1 0x1 0 True
1 False 2 0x2
1 1x0 2 0x0
1 False 1 0x1
1 0x2 1 0x0
1 1x2 1 0x2
¥sum 1x1
¥carry 1x2
```

Synthesized netlist

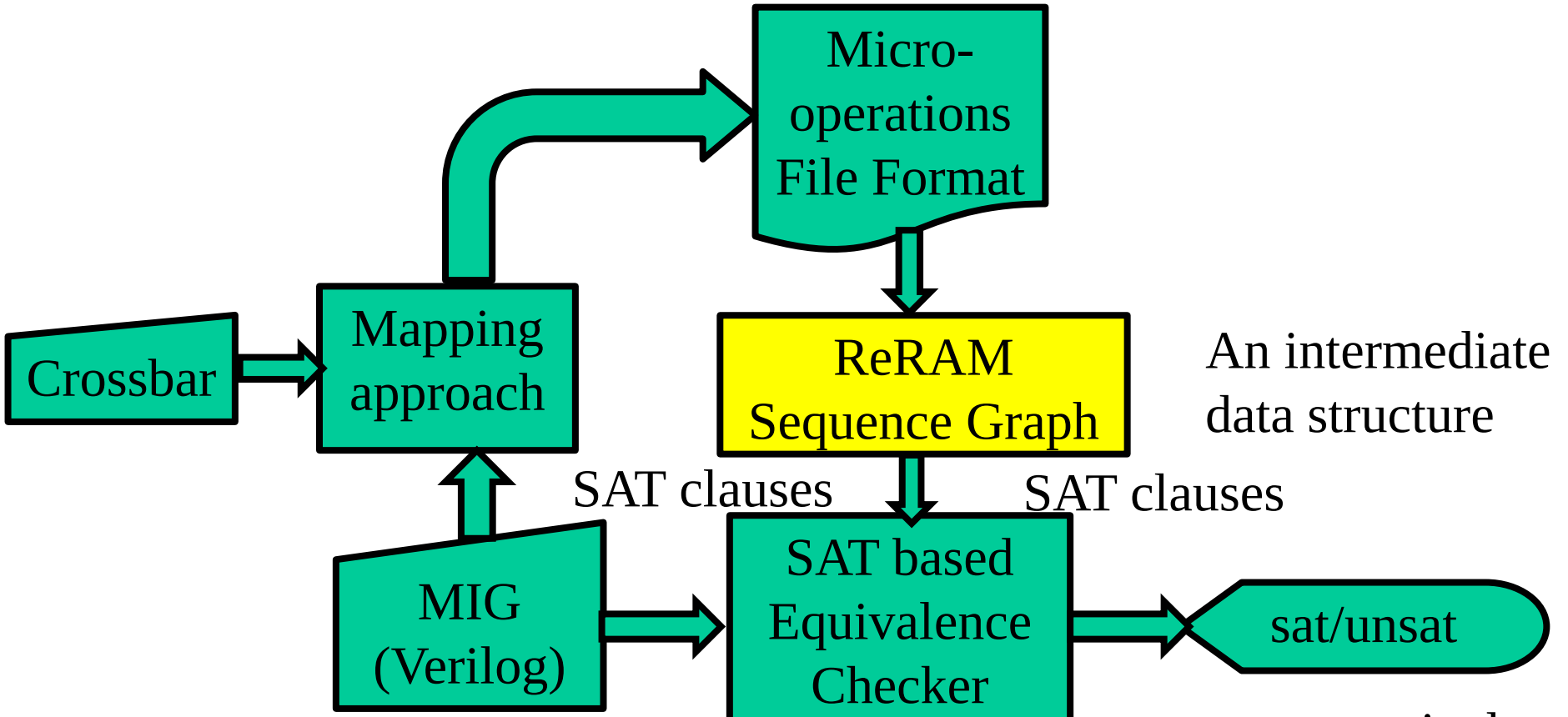
Possible Approach

- Manual inspection
  - impossible for large and complex circuits
- Simulation-based
  - limited for a subset of input combinations
- **Equivalence checking** (necessary)
  - Can handle complex in-memory designs

Equivalence Check

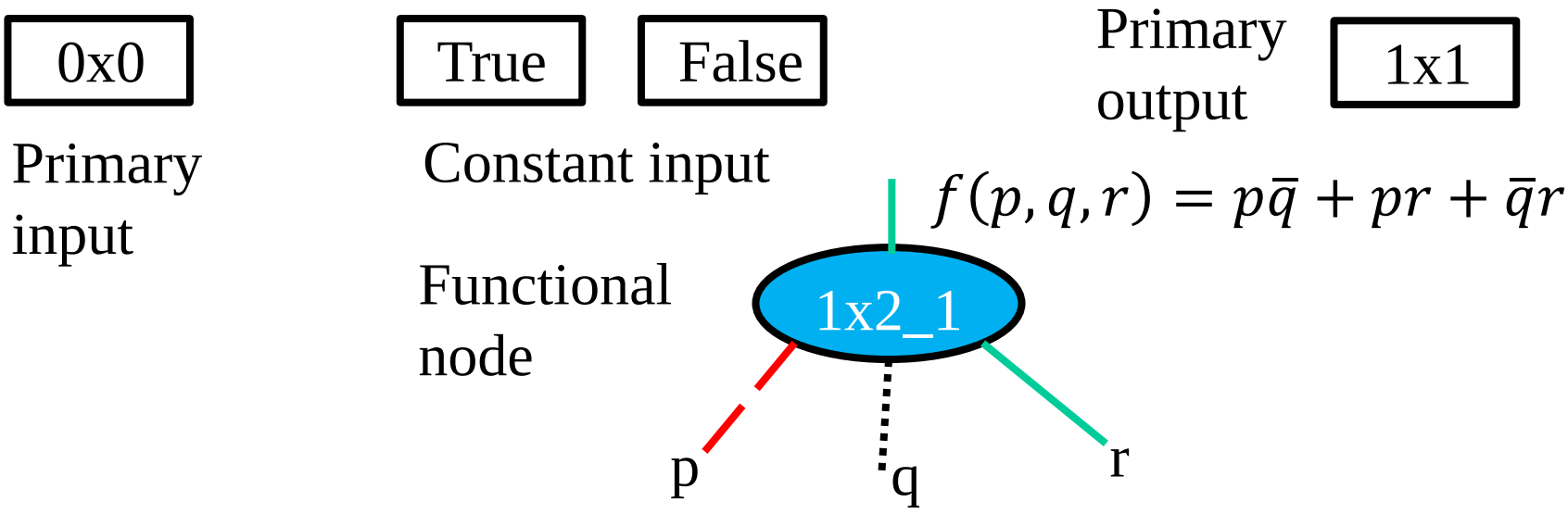


Proposed Approach



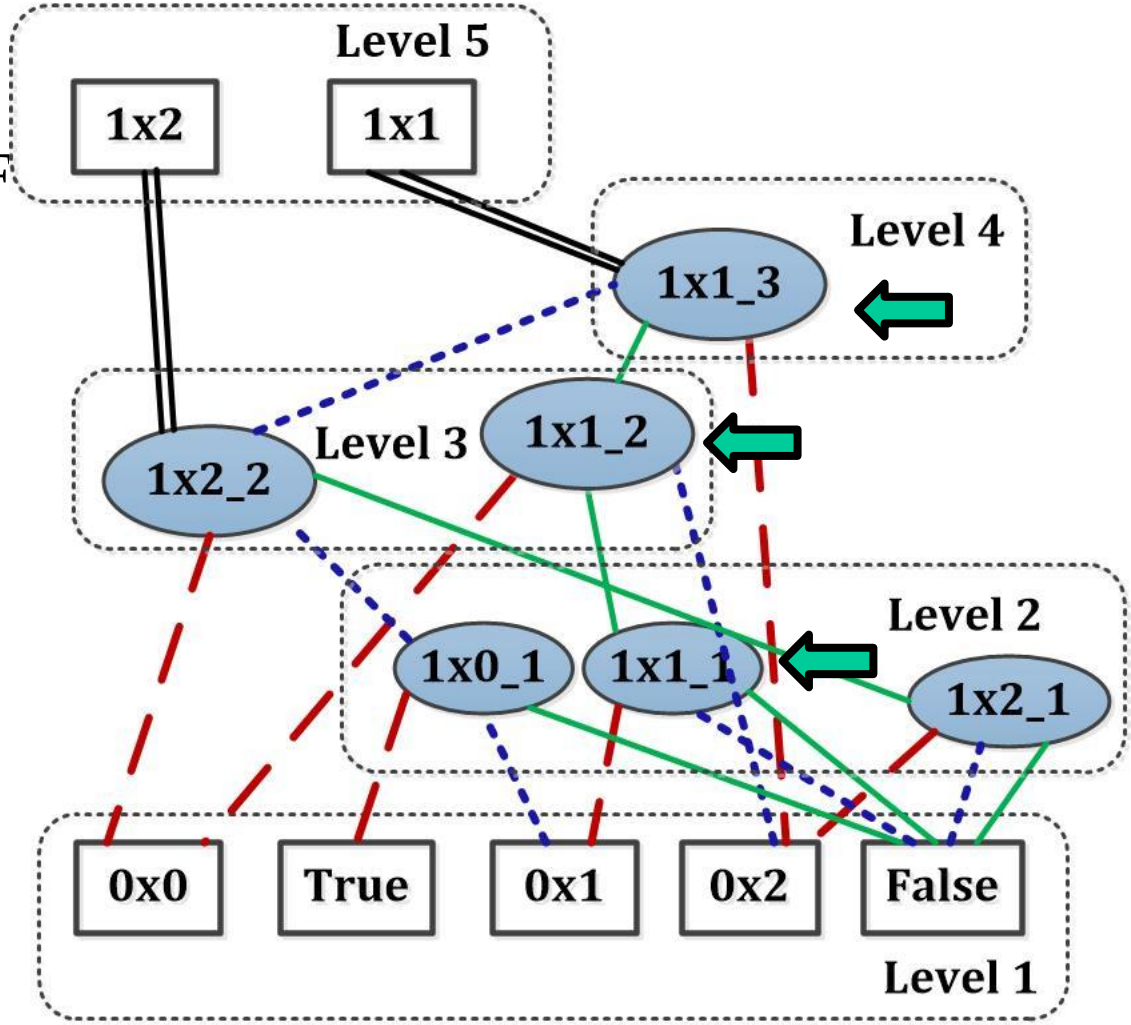
ReSG

- ReRAM sequence graph
- Directed acyclic graph
- Four types of nodes

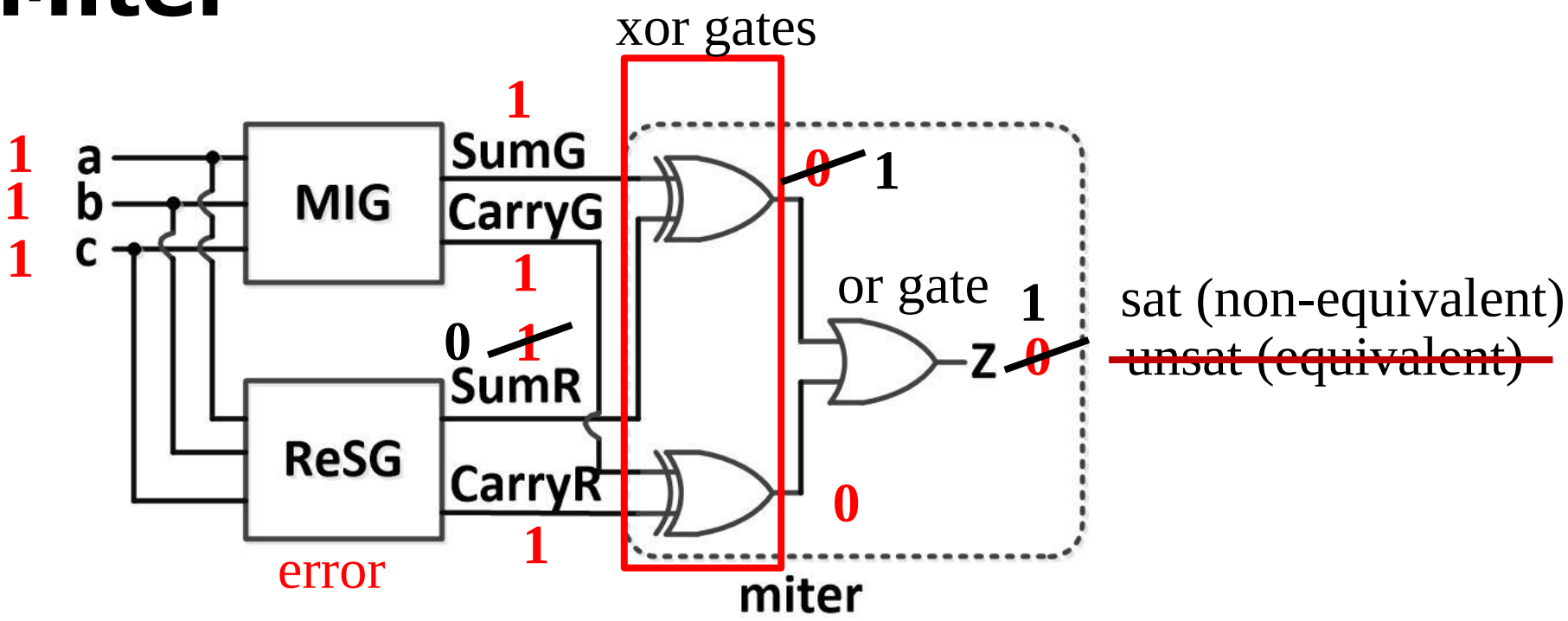


Example

0 0 ¥a 1 ¥b 2 ¥c
1 True 0 False 1 False 2 F
1 0x1 0 True
1 False 2 0x2
1 1x0 2 0x0
1 False 1 0x1
1 0x2 1 0x0
1 1x2 1 0x2
¥sum 1x1
¥carry 1x2



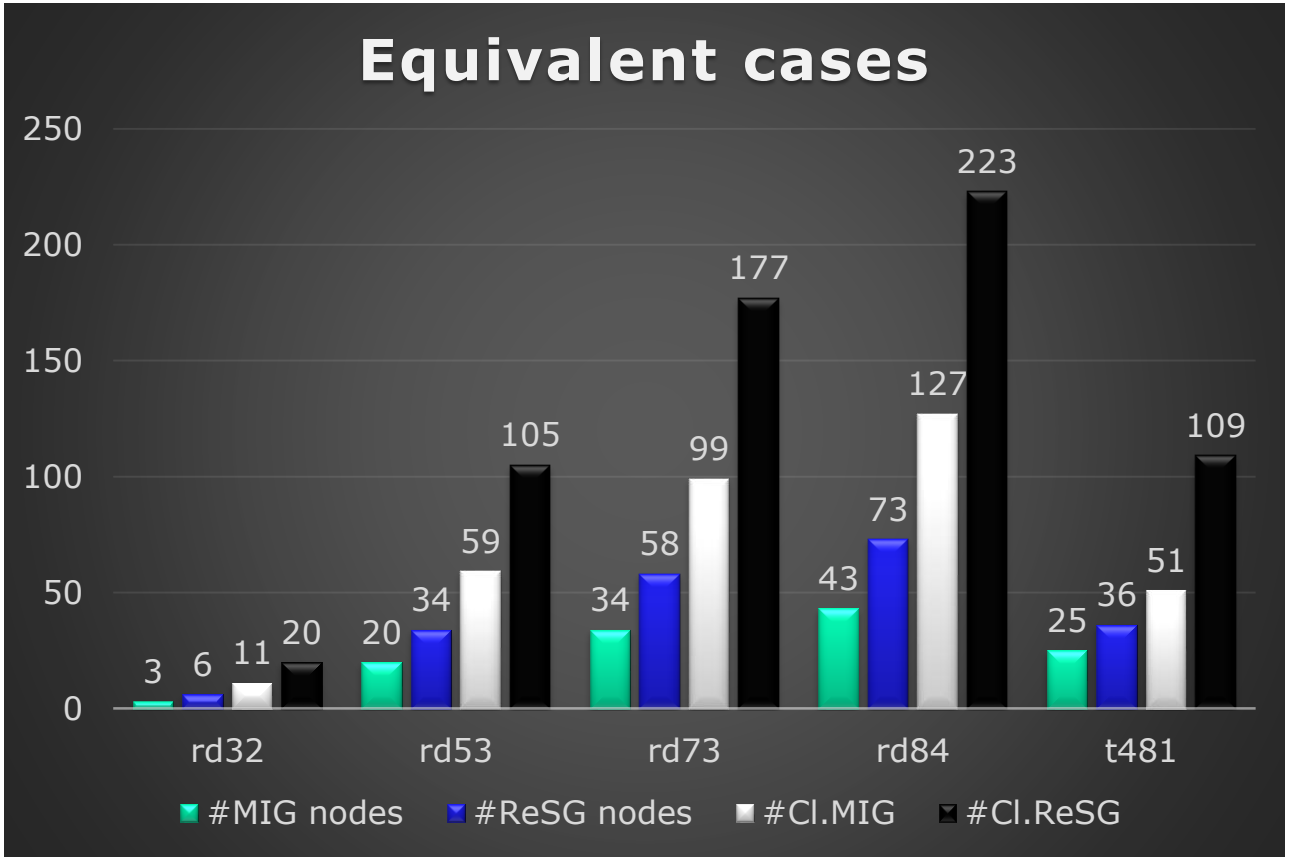
Miter


xor gates
1
1
a
1
b
1
c
MIG
SumG
CarryG
1
0
1
SumR
ReSG
CarryR
error
1
0
1
or gate
1
Z
0
sat (non-equivalent)
unsat (equivalent)
0
miter

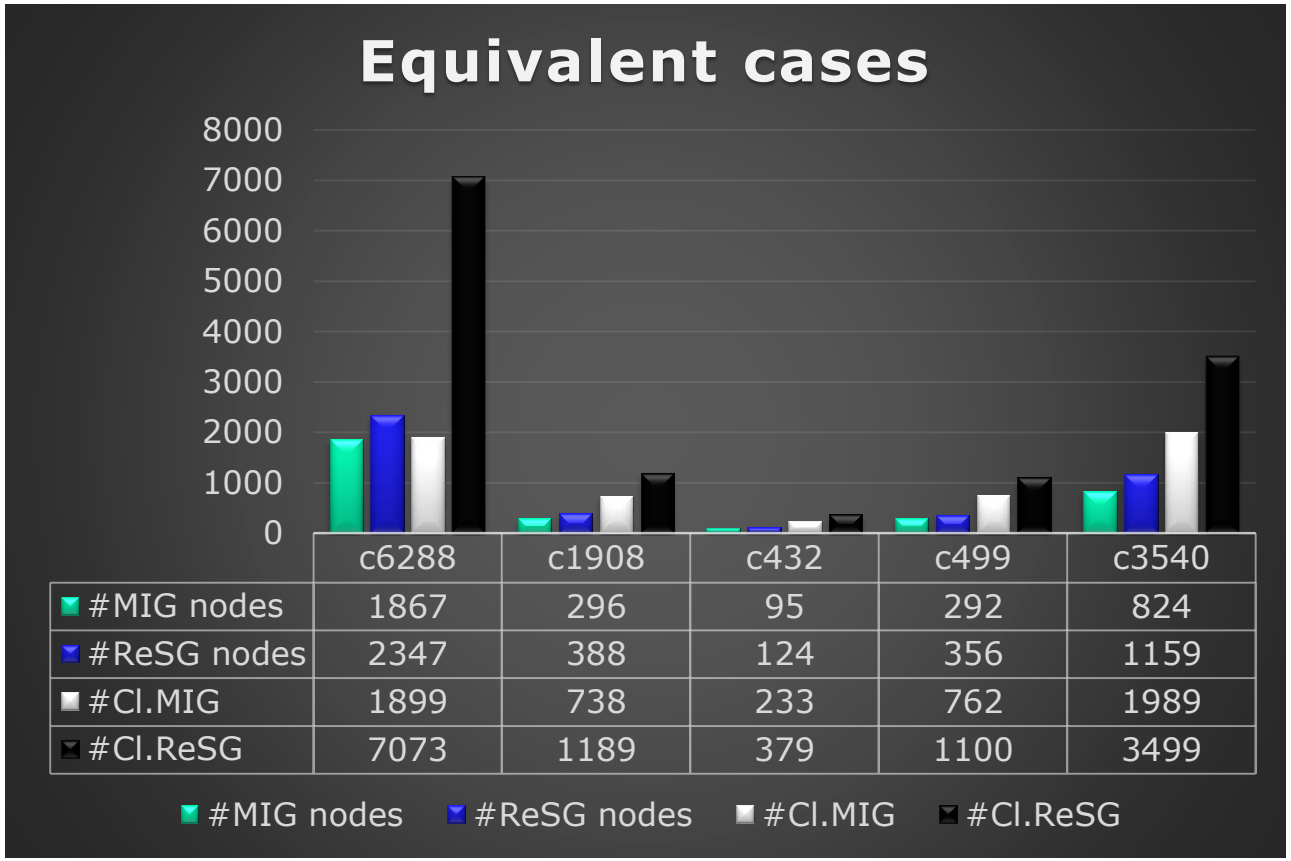
Results

- ISCAS-85 and IWLS benchmarks
- MIG representations
  - Generation of SAT clauses
  - CNF: $f(a,b,c) = \overline{(\bar{a}+\bar{b})(\bar{a}+\bar{c})(\bar{b}+\bar{c})}$
- Micro-operations file format
  - ReSG formulation
  - SAT clause generation
  - CNF: $f(p,q,r) = \overline{(\bar{p}+q)(\bar{p}+\bar{r})(q+\bar{r})}$

IWLS

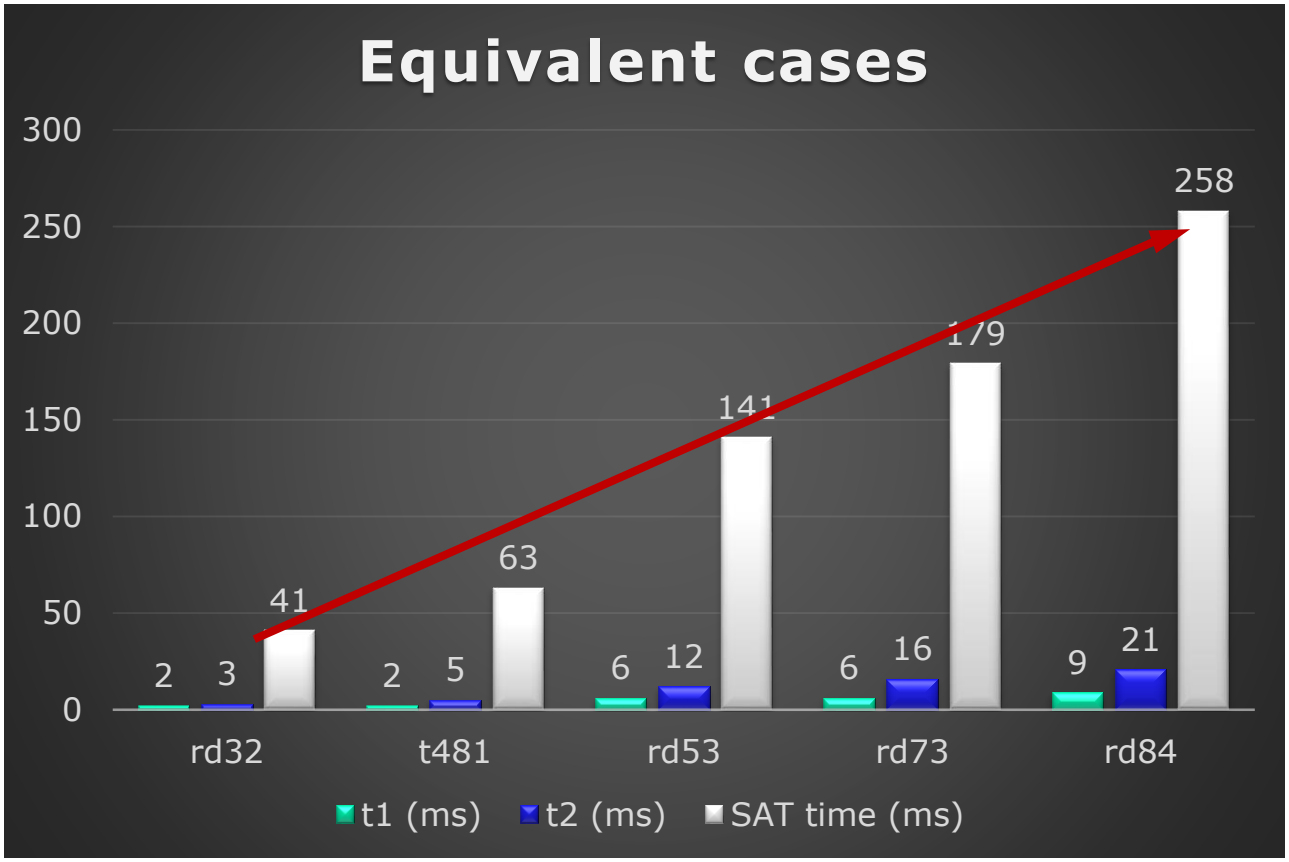

Equivalent cases
250
200
150
100
50
0
3
6
11
20
20
34
59
105
34
58
99
177
43
73
127
223
25
36
51
109
rd32
rd53
rd73
rd84
t481
#MIG nodes
#ReSG nodes
#Cl.MIG
#Cl.ReSG

ISCAS-85

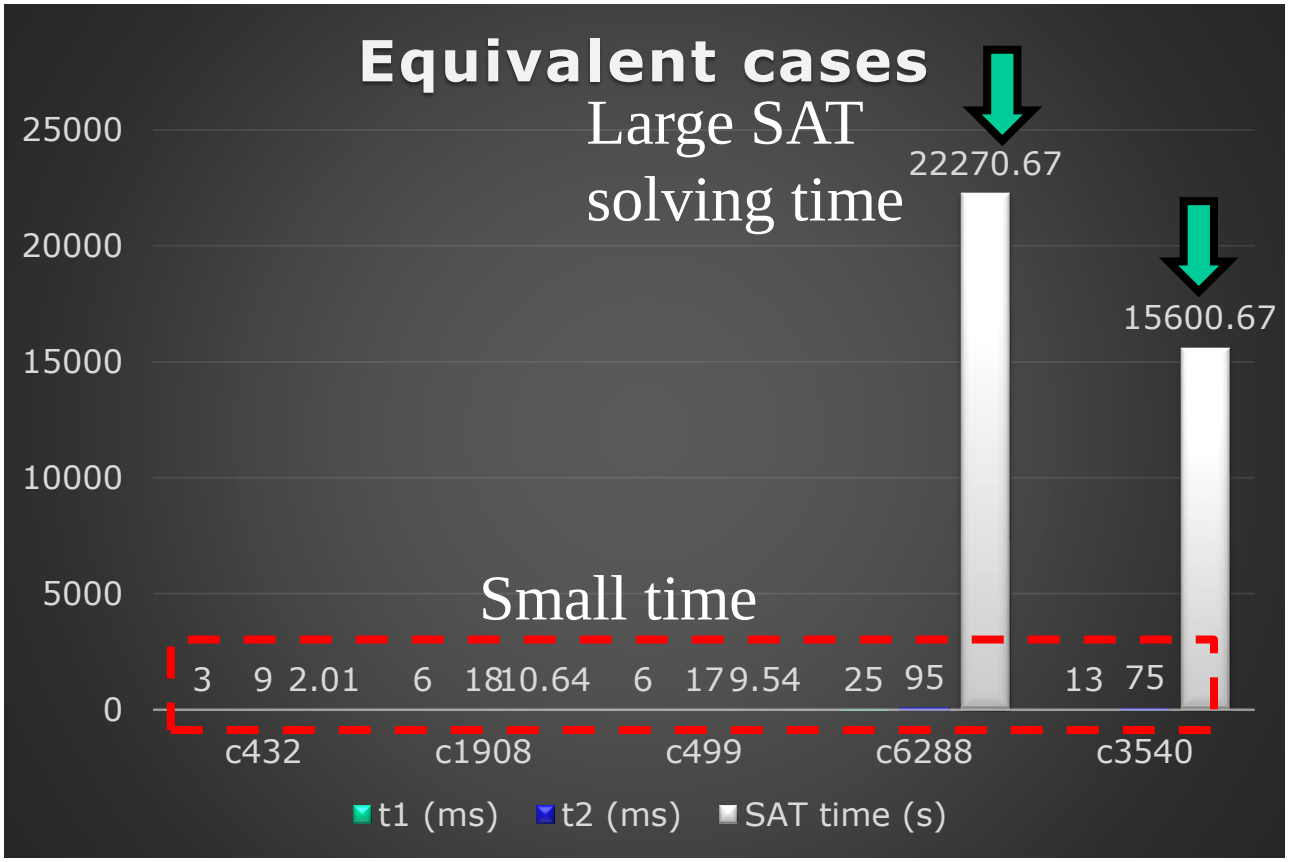


	c6288	c1908	c432	c499	c3540
#MIG nodes	1867	296	95	292	824
#ReSG nodes	2347	388	124	356	1159
#Cl.MIG	1899	738	233	762	1989
#Cl.ReSG	7073	1189	379	1100	3499

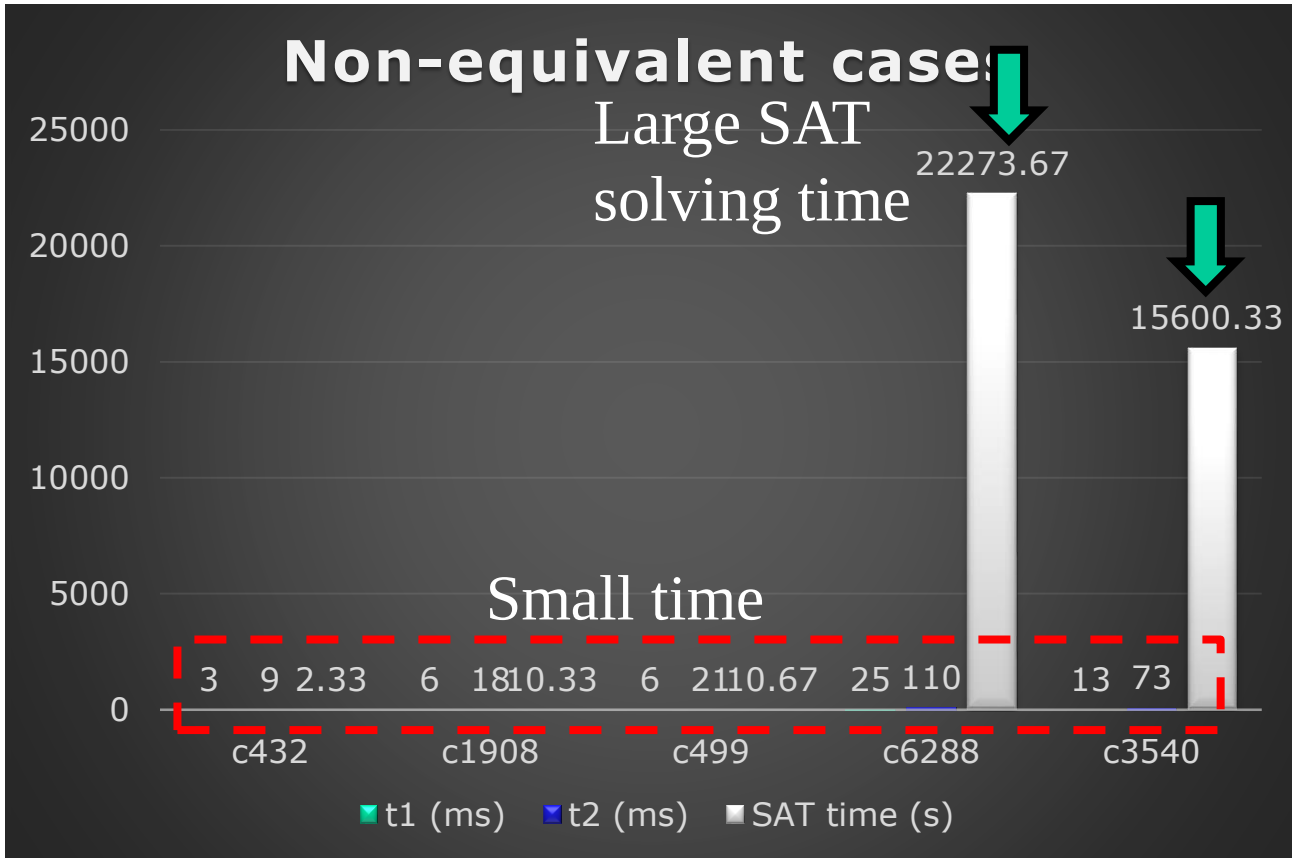
SAT Solver Time



SAT Solver Time



Non-equivalence



Conclusion

- Automatic verification flow
- Equivalence checking between MIG and micro-operations file format
- ReRAM sequence graph
- Future work
  - Equivalence checking for MAGIC based design

Universität Bremen

German Research Center for Artificial Intelligence

Automated Equivalence Checking Method for Majority based In-Memory Computing on ReRAM Crossbars

Arighna Deb[1], Kamalika Datta[2], Muhammad Hassan[2,4], Saeideh Shirinzadeh[2,3], Rolf Drechsler[2,4]

[1]KIIT University, Bhubaneswar, India
[2]DFKI GmbH, Bremen, Germany
[3]Fraunhofer ISI, Karlsruhe, Germany
[4]University of Bremen, Bremen, Germany

muhammad.hassan@dfki.de

Bundesministerium für Bildung und Forschung
16ME0117
AUTOASSERT