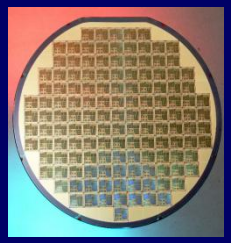


TECHNION – ISRAEL INSTITUTE OF TECHNOLOGY
VLSI SYSTEMS RESEARCH CENTER
VLSI LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE

Manipulating Decision Methods of Modern SAT Solvers for Bounded Model Checking

Authors:

Yael Weinberg

Osnat Tal

Instructor:

Avi Yadgar

Outline

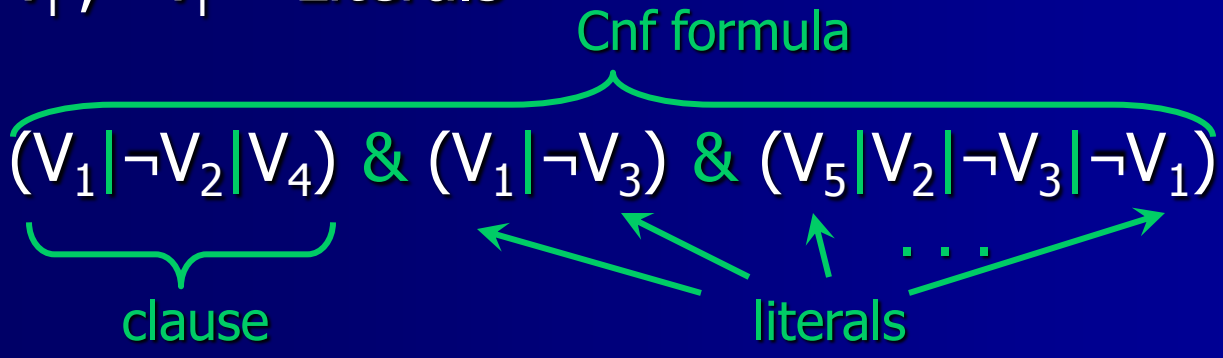
- Background
  - List of terms and symbols
  - CNF,SAT problems, BMC
- Motivation
- Project Target
- Our Algorithm
- Example
- Results

Background: Terms

- *CNF - Conjunctive Normal Form.*
- *SAT - Boolean Satisfiability Problem.*
- *Sat Solver – a verification tool for SAT problem.*
- *VSIDS - Variable State Independent Decaying Sum algorithm.*
- *ZChaff – The original application, developed at Princeton University.*
- *BMC – Bounded Model Checking.*
- *DFS – Depth First Search algorithm.*

Background: CNF formulas

- CNF - Conjunctive Normal Form
- $V_1, V_2, V_3 \ldots V_n$ – Boolean Variables
- V_i , $\neg V_i$ – Literals



- Variable Assignment – setting value for each of the variables.
  For example, $V_1 \leftarrow 0, V_2 \leftarrow 1, V_3 \leftarrow 0, V_4 \leftarrow 1, V_5 \leftarrow 0$.

Background: Sat

- Satisfying Assignment - A variable assignment which sets the formula's value to 1 (TRUE).
- For the given example, one of the possible assignments is $v_1 \leftarrow 1, v_5 \leftarrow 1, v_2 \leftarrow 0, v_3 \leftarrow 0, v_4 \leftarrow 0$.
- Sat (Boolean Satisfiability Problem) :
  Given a Boolean formula, determine whether exists a satisfying variable assignment.
- If exists, the formula is SAT, otherwise – UNSAT.
- SAT is one of the central NP-complete problems.

Background: Sat Solvers

- Sat Solver – a verification tool. Determines if a given cnf formula is SAT, and if so, returns a satisfying assignment.
- Most of the solvers implement Davis-Putnam Backtrack Search.
- One of the properties that differ Sat Solvers is the decision method - The heuristic for choosing the variable to be assigned in each iteration.

ZChaff – The original application

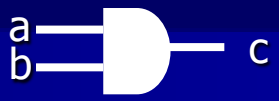
- A Sat solver, developed at Princeton University. http://www.princeton.edu/~chaff/
- A variation of Davis-Putnam Backtrack Search.
- Decision method: VSIDS - Variable State Independent Decaying Sum.
- Variables are sorted by their score – sum of all occurrences in the formula, for each polarity.
- At each decision, the (unassigned) variable and polarity with the highest score is chosen to be assigned.

Hardware Verification

- Given:
  - A circuit specification
    (inputs, logical gates, and state variables)
  - Initial constraints
  - Invariants for correctness (Assertions)
- Target:
  Determine whether a bug exists.
  - If so, than exists a program flow of k cycles, which falsifies the assertions.
  - This is done by bounded model checking.

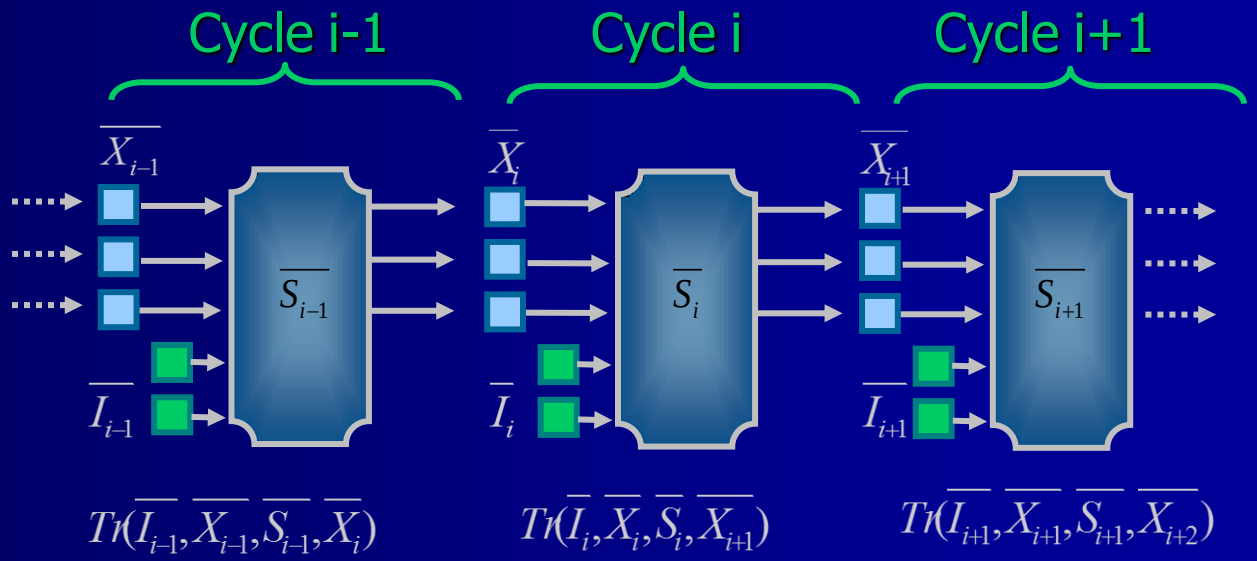
BMC – Bounded Model Checking

- Gate translation
- Circuit translation
- Circuit replication
- Final translation



Translation

$(\neg a|\neg b|c)\ \&\ (a|\neg c)\ \&\ (b|\neg c)$



$\overline{Ii}$ - inputs

$\overline{Xi}$ - state variables

$\overline{Si}$ - logical gates

BMC – cont.

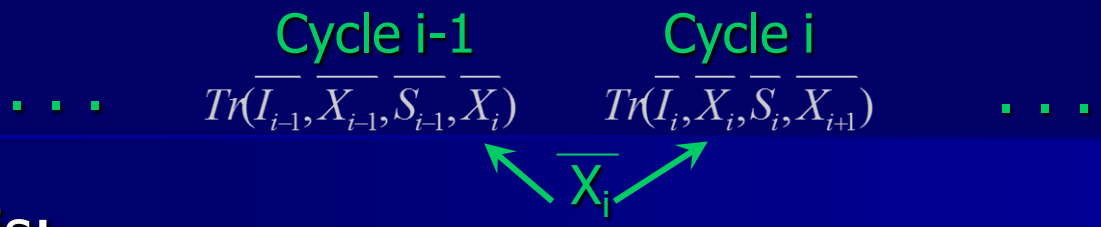
- For a given k, translate the specification to a cnf formula representing a program flow of k cycles.
- Concatenate to it the cnf formula representing the negation of the assertion.
- Increment k from 0 to a bounded limit. Each step, run Sat solver on the representing cnf formula.
- If it is SAT for a specific k – there is a bug! (of k length)

Initial constrains → $I_0(\overline{X_0}) \wedge$

Cycles transactions → $Tr(\overline{I_0}, \overline{X_0}, \overline{S_0}, \overline{X_1}) \wedge Tr(\overline{I_1}, \overline{X_1}, \overline{S_1}, \overline{X_2}) \wedge \ldots \wedge Tr(\overline{I_{k-1}}, \overline{X_{k-1}}, \overline{S_{k-1}}, \overline{X_k}) \wedge$

Invariant negation → $\neg\varphi(\overline{X_0}, \ldots, \overline{X_k}, \overline{I_0}, \ldots, \overline{I_{k-1}}, \overline{S_0}, \ldots, \overline{S_{k-1}})$

Motivation



Diagnosis:

- Clauses of different cycles only share state variables between them. This happens for adjacent cycles only. Meaning, cycle i-1 and cycle i only share $\overline{X_i}$ variables.
- If we assign all $\overline{X_i}$ state variables first, we will divide the entire formula into two independent formulas (with no common variables).
- Why stop at one partition? We shall continue dividing the formula as long as possible.
- A significant improvement for NP Complete problems.

Project target

- Associate each state variable to its cycle.
- Determine the optimized partition order (based on cycle index)
- Modify Decision method to support the suggested assignment order.
- Project Constraints:
  - Perform required adjustments on Zchaff
  - Implementation in C++
  - For each cycle, the original Decision heuristics (VSIDS) should be reserved.

Our Algorithm - basics

- State variables are organized by cycles.
- Each cycle keeps its state variables sorted by their score.
- Partition order is determined according to dfs search, as follows:

4	3	5	2	7	6	8	1	11	10	12	9	14	13	15	Partition order
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Cycle index

- =>The optimized partition order will be:
  cycles 7,3,1,0,2,5,4,6,11,9,8,10,13,12,14

Our Algorithm - decision heuristic

- Traverse cycles according to partition order.
- Each iteration, choose the unassigned state variable with the highest score of the current cycle .
- After all state variables were assigned (end of partitions), assign all remainder variables, chosen by their frequency in the formula.

Example – assignment order

x_i , y_i - state variables

a_i - input or logical gates

$$(x_0)\wedge$$
$$(x_0 \vee a_0 \vee x_1)\wedge(a_0 \vee \neg y_0 \vee x_1)\wedge$$
$$(x_1 \vee a_1 \vee x_2)\wedge(a_1 \vee \neg y_1 \vee x_2)\wedge$$
$$(x_2 \vee a_2 \vee x_3)\wedge(a_2 \vee \neg y_2 \vee x_3)\wedge$$
$$(x_3 \vee a_3 \vee x_4)\wedge(a_3 \vee \neg y_3 \vee x_4)\wedge$$
$$(x_4)$$

Original

Variable	Score
X_1	3
X_3	3
X_2	3
X_4	3
a_2	2
a_0	2
x_0	2
a_3	2
a_1	2
y_1	1
Y_2	1
Y_0	1
y_3	1
y_4	0

Our

Variable	Score	
X_2	2	Cycle 2
y_2	1	
X_0	3	Cycle 0
y_0	1	
X_1	3	Cycle 1
y_1	1	
X_3	3	Cycle 3
y_3	1	
X_4	3	Cycle 4
y_4	0	
a_2	2	Non state Variables
a_1	2	
a_3	2	
a_0	2	

Results:

Input file S35932:

- 35 inputs
- 320 outputs
- 1728 D-type flipflops
- 3861 inverters
- 12204 gates (4032 ANDs + 7020 NANDs + 1152 ORs + 0 NORs)

AG ((WX491|WX503|WX537)&(WX11177|DATA_0_31|!WX503|WX539)& (WX535|!WX11179|WX8287))

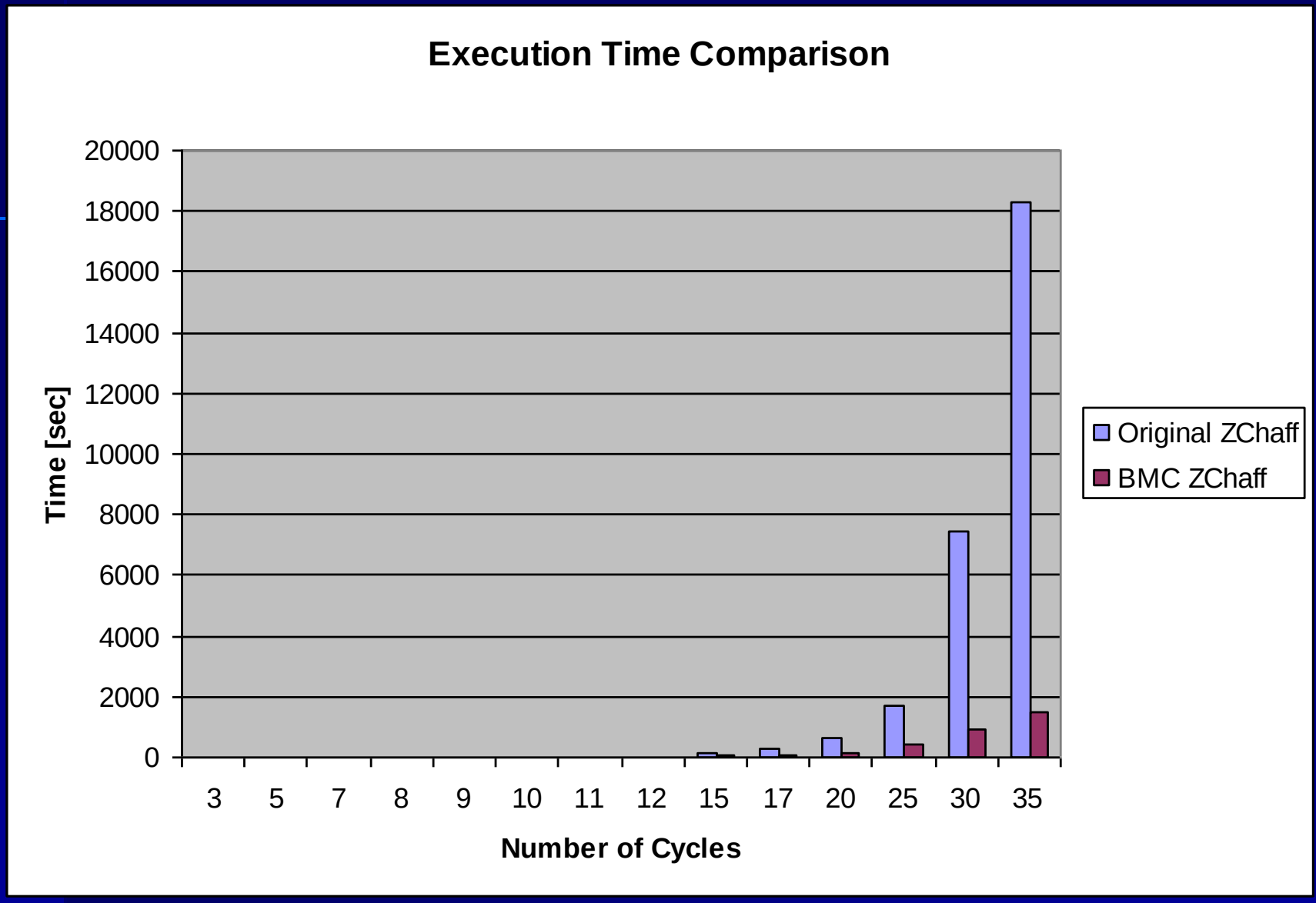
Input file B17:

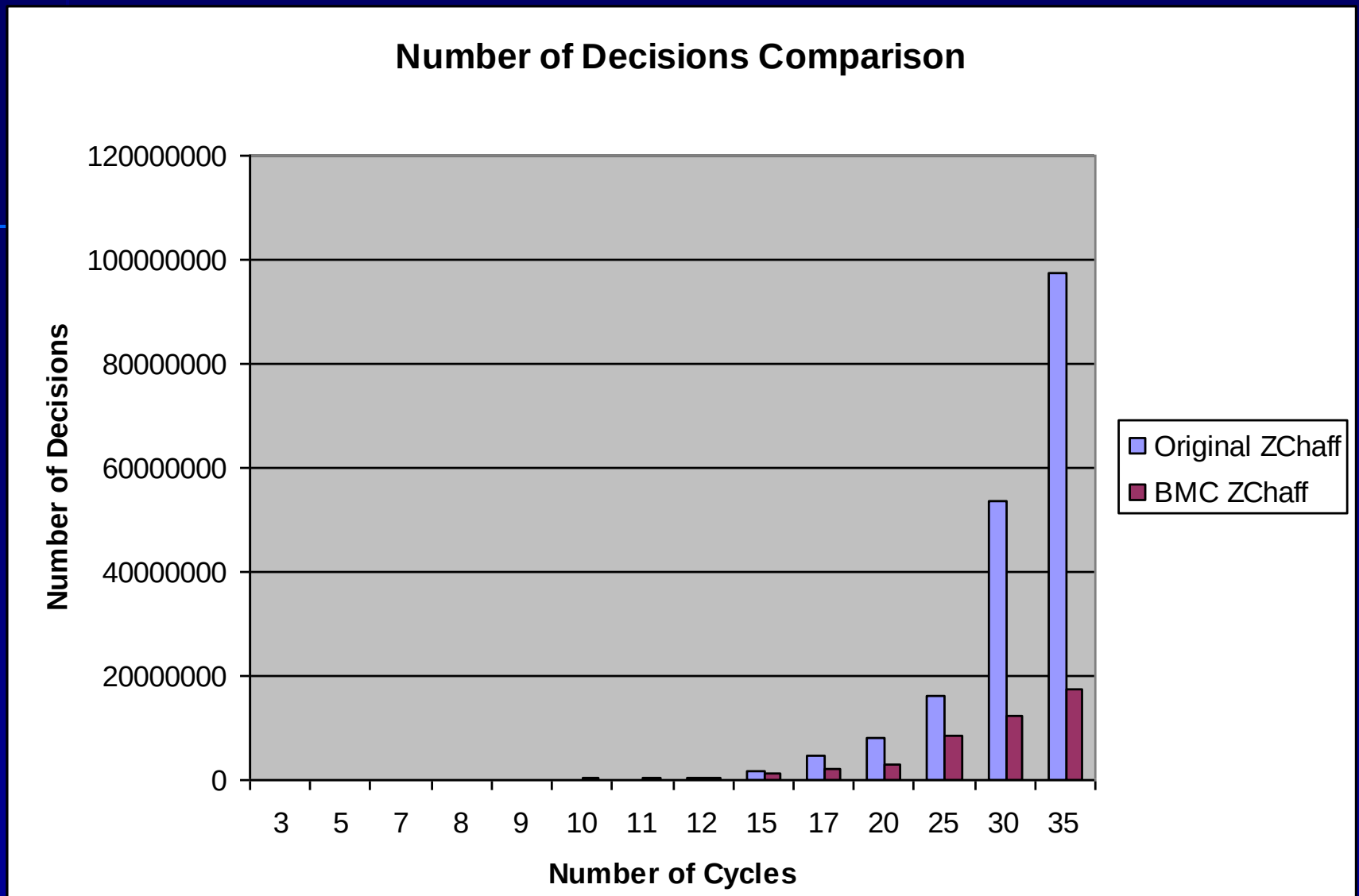
- 37 inputs
- 97 outputs
- 1415 D-type flipflops
- 4474 inverters
- 27852 gates (4054 and, 21815 nand, 299 or, 135 nor, 4474 not)

AG ((BUF1_REG_12_ | !BUF1_REG_2_ |P3_ADDRESS_REG_24_) &
(!BUF1_REG_14_ | P1_INSTQUEUE_REG_8__5_| P2_DATAWIDTH_REG_16_ |
!P1_LWORD_REG_1_ | !P1_EBX_REG_0_ | BUF1_REG_2_))

S35932:

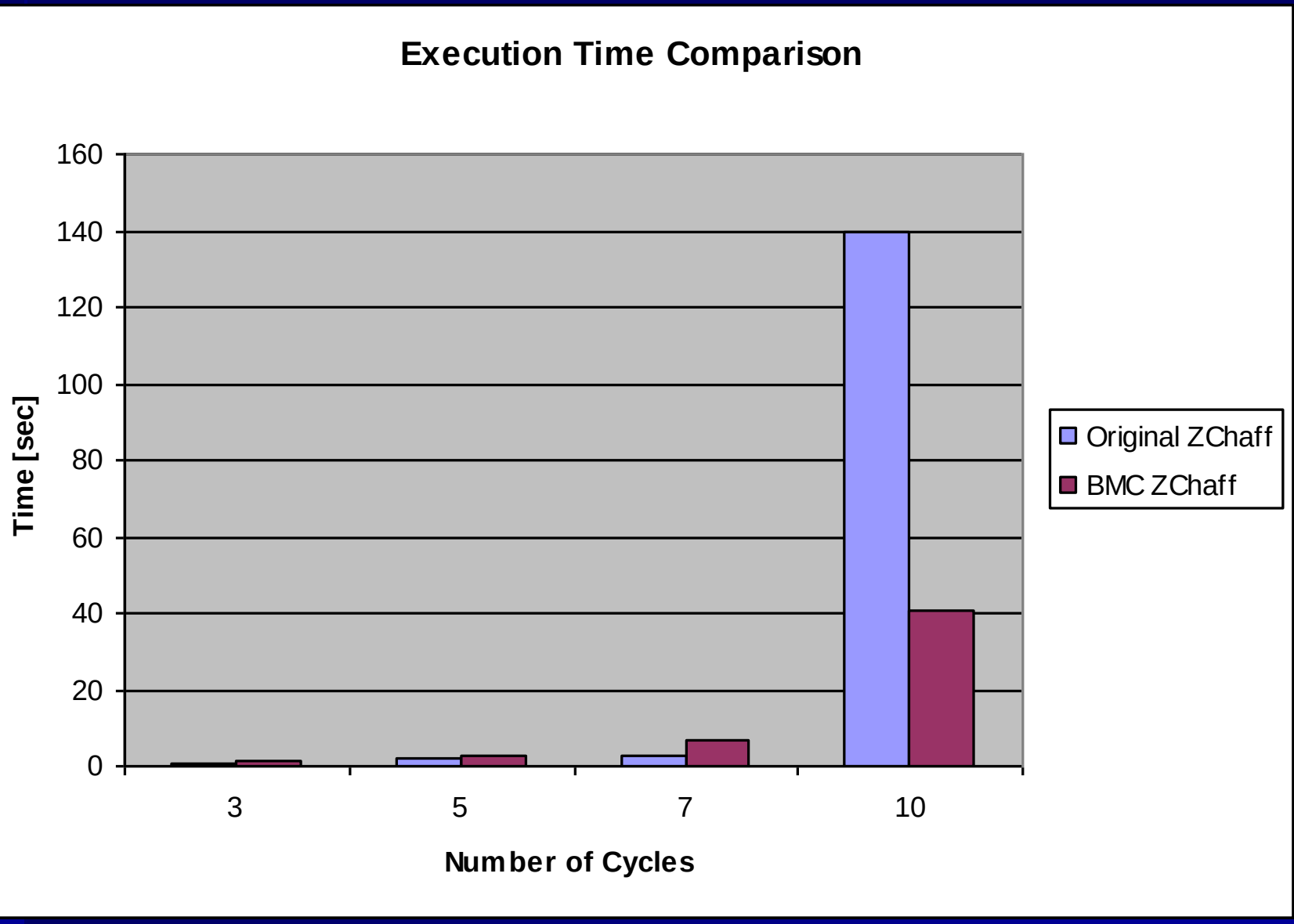
CNF File			Original Zchaff		BMC ZChaff	
# cycles	# variables	# clauses	Time	# decisions	Time	# decisions
3	57010	160356	0.112007	1861	0.120008	5300
5	93864	267258	0.436027	12817	0.272017	11822
7	130718	374160	1.19607	26479	0.996062	47218
8	149145	427611	2.14013	45942	3.2482	176903
9	167572	481062	2.50016	47540	2.04813	129363
10	185999	534513	4.30427	73455	10.9487	531311
11	204426	587964	4.7323	76168	11.2727	593630
12	222853	641415	19.7852	397466	9.6286	425772
15	278134	801768	120.624	1912741	50.3311	1218501
17	314988	908670	318.58	4482253	95.486	2023768
20	370269	1069023	658.849	7981844	145.013	2940430
25	462404	1336278	1716.34	16162845	414.454	8415327
30	554539	1603533	7438.46	53775854	906.857	12226362
35	646674	1870788	18290.2	97651404	1476.6	17389262


Execution Time Comparison
Time [sec]
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
3
5
7
8
9
10
11
12
15
17
20
25
30
35
Number of Cycles
Original ZChaff
BMC ZChaff


Number of Decisions Comparison
120000000
100000000
80000000
60000000
40000000
20000000
0
Number of Decisions
3
5
7
8
9
10
11
12
15
17
20
25
30
35
Number of Cycles
Original ZChaff
BMC ZChaff

B17:

CNF File			Original Zchaff		BMC ZChaff	
# cycles	# variables	# clauses	Time	# decisions	Time	# decisions
3	84596	801767	0.404025	4801	1.07607	27999
5	140050	1336277	2.35215	13207	2.78817	48409
7	195504	1870787	2.95618	13190	6.48441	60241
10	278685	2672552	139.645	266358	40.5825	299623


Execution Time Comparison
160
140
120
100
80
60
40
20
0
Time [sec]
3
5
7
10
Number of Cycles
Original ZChaff
BMC ZChaff

Thank you