Journal of Circuits, Systems, and Computers
Vol. 20, No. 8 (2011) 1605−1618
© World Scientific Publishing Company
DOI: 10.1142/S0218126611007980

USING SAT-BASED TECHNIQUES IN LOW POWER STATE ASSIGNMENT[*]

ASSIM SAGAHYROON[†] and FADI A. ALOUL[‡]

Department of Computer Science and Engineering,
American University of Sharjah,
PO Box 26666, Sharjah, UAE
[†]*asagahyroon@aus.edu*
[‡]*faloul@aus.edu*

ALEXANDER SUDNITSON

Department of Computer Engineering,
Tallinn University of Technology, Estonia
alsu@cc.ttu.ee

Received 15 February 2010
Accepted 27 June 2011

Power consumption of synchronous sequential circuits can be reduced by careful encoding of the states of the circuit. The idea is to reduce the average number of bit changes per state transition by finding an optimal state assignment. This state assignment problem is NP-hard, and existing techniques rely mainly on heuristic-based methods. The primary goal of this work is to assess the suitability of using complete advanced Boolean Satisfiability and Integer Linear Programming (ILP) methods in finding an optimized solution. We formulate the problem as a 0-1 ILP instance with power minimization being the objective. Using generic and commercial solvers, the proposed approach was tested on sample circuits from the MCNC benchmark suite. Furthermore, in an effort to accelerate the search process, circuits were checked for symmetries and symmetry breaking predicates were added whenever applicable. The experimental results provide a pragmatic insight into the problem and basis for further exploration.

Keywords: State assignment; power; integer linear programming; Boolean Satisfiability.

1. Introduction

With the ever increasing integration scale, power consumption has emerged as a major design constraint for integrated circuits. During logic synthesis of sequential circuits, i.e., finite state machines (FSMs), assigning binary codes to each state in the circuit is a critical step in the low power design of the circuit. The state encoding problem for low power has been explored by a number of researchers. Recent work

[*]This paper was recommended by Regional Editor Krishna Shenai.

has attempted to address this issue using various techniques. In Ref. 1, using a probabilistic description of the circuit, a state assignment algorithm that minimizes the Boolean distance between the codes of the state with high transition probability was proposed. The primary objective was to reduce the average switching activity of the input and output state variables by minimizing the number of bit changes during state transition. Symbolic methods for reencoding a circuit to reduce the dissipated power are discussed in Ref. 2. The authors experimented with two encoding strategies. One based on recursive weighted no-bipartite matching, and one on recursive mincut bipartitioning. The authors conclude that even though the results are promising, their synthesis procedure needs refinements in both circuit transformation and the power estimation phases. Encoding of states for a given user-specified input sequence was addressed in Ref. 3. Simulation was used to determine the relative frequency for all state transitions. By defining and using a register switching rate as a cost, the authors then used simulated annealing to solve the state assignment problem. In Ref. 4, given an FSM description and the input probabilities, the total state transition probabilities for each edge in a state transition graph of the circuit are computed first. This is done by modeling the FSM as a Markov chain. A cost function that is the summation of the products of distinct code words with the Hamming distance between any two adjacent states is used to measure the effectiveness of the assignments. Similar to Ref. 1, the goal is to assign codewords with minimum Hamming distances to states with higher transition probabilities. A spanning tree-based encoding approach was tested in Ref. 5. The assignment problem was formulated as a hypercube embedding problem, where the embedding process is directed by a maximum spanning tree of the attraction graph of the FSM. In Ref. 6, heuristic techniques were used to visit the state transition graph of the circuit and assign a priority to the symbolic states. Next, an encoding technique that follows the priority established in the first part is used to assign binary codes to the states. As assignment method that utilized dynamic loop information extracted from FSM profiling data was presented in Ref. 7. The authors then experimented with three different loop-based state assignment algorithms, depth-first search, loop-based depth first and per state encoding. An m-block partitioning technique for the state assignment to reduce the number of feedback cycles and keep low switching activities among state variables is discussed in Ref. 8. The objective was both, to improve power consumption and testability of the circuit. Reengineering (targeting low power) a FSM by constructing a functionally equivalent but topologically different design based on an optimization objective was presented in Ref. 9. The authors argue that this will allow the exploration of a larger solution space making it possible to obtain solutions better than the optimal ones for the original circuit. They used genetic algorithms and heuristics to reengineer existing low power state encoding procedures with favorable results.

The last decade have seen a remarkable growth in the use of Boolean Satisfiability (SAT) models and algorithms for solving various problems in Electronic Design

Automation (EDA). This is mainly due to the fact that SAT algorithms have seen tremendous improvements in the last few years, allowing larger problem instances to be solved in different applications domains.[10–13] Such applications include formal verification,[14] FPGA routing,[15] global routing,[16] logic synthesis,[17] and power optimization.[18]

SAT solvers have traditionally been used to solve *decision* problems. Given a set of Boolean variables and constraints expressed in products-of-sum form (also known as *conjunctive normal form* (CNF)), the goal is to identify a variable assignment that will satisfy all constraints in the problem or prove that no such assignment exists. Recently, SAT solvers have been extended to handle Pseudo-Boolean (PB) constraints.[16,19–22] PB constraints are more expressive and can replace an exponential number of CNF constraints. Another key advantage of PB constraints is the ability to express *optimization* problems which are traditionally handled as integer linear programming (ILP) problems. Hence, SAT solvers can now handle both decision and optimization problems and compete with the best available generic ILP solvers such as IBM-ILOG CPLEX.[23]

In this paper, we propose using SAT-based and generic ILP solvers to handle the state assignment problem. We formulate the problem as an optimization instance and use SAT-based ILP solvers to find a solution. The problem is also solved using generic ILP solvers, e.g., CPLEX,[23] for performance comparison. Interestingly, presented empirical results show that SAT-based ILP solvers outperform generic commercial ILP solvers when solving such instances.

Previous work has shown that breaking symmetries in SAT 0-1 ILP formulas effectively prunes the search space and can lead to significant runtime speedups.[1] The idea is that breaking symmetries prevent symmetric images of search paths from being searched, thus pruning the search tree.[24,25] Since industrial instances, in general, have some structure, they are likely to have symmetries.

In this paper, we statically detect and break the symmetries in the generated SAT 0-1 ILP instances and evaluate the advantage of that on SAT-based and generic ILP solvers. Our results indicate the presence of symmetries in the tested instances. Furthermore, as suggested by previous work,[26] SAT-based ILP solvers perform better on instances with broken symmetries while generic ILP solvers show mixed performance.

The rest of the paper is organized as follows: Sec. 2 provides background information on the state assignment problem and Boolean SAT; Sec. 3 explains how to formulate the state assignment problem as a SAT 0-1 ILP problem. Section 4 gives a summary of the experimental results and the paper is concluded in Sec. 5.

2. Background

In this section, we review the state assignment problem and we introduce some key concepts related to Boolean SAT and symmetry breaking. This will serve in clarifying our proposed approach to solve the state assignment problem.

2.1. *The state transition graph*

The state assignment problem entails the codification of states in a FSM, and is a known NP-complete problem.[27] In Ref. 1, a brief attempt was made to formulate the state encoding as an ILP problem. However, at the time there were no commercial or state-of-the-art solvers available and the authors only briefly discussed a framework to formulate the problem.

In this work, prior to formulating the problem as an ILP instance, the state transition graph (STG) for each benchmark circuit is developed. That is, given the FSM description and the input probabilities; we compute the transition probabilities for each edge in the STG, by modeling the FSM as a Markov chain. Thereafter, all unreachable states and self-loops are eliminated from the graph. The STG is then transformed into an undirected graph converting all multiple edges into a single undirected edge with weights that equals to the sum of the directed edge probabilities.[1,6] Thus, the weights on the edges are proportional to the total probability of transition between every two connected states.

2.2. *Boolean SAT*

Boolean SAT is often used as the underlying model in the field of computer aided designs of integrated circuits. A number of SAT solvers have been proposed and implemented.[10–13] These solvers employ powerful algorithms that are sufficiently efficient to deal with large-scale SAT problems that typically arise in the design automation domain. Most of these algorithms claim competitive results in runtime efficiency and robustness.

In SAT, given a formula f, the objective is to identify an assignment to a set of Boolean variables that will satisfy a set of constraints. If an assignment is found, it is known as a satisfying assignment, and the formula is called *satisfiable*. Otherwise if an assignment does not exist, the formula is called *unsatisfiable*. The constraints are typically expressed in CNF. In CNF, the formula consists of the conjunction (AND) of m clauses $\omega_1, \ldots, \omega_m$ each of which consists of the disjunction (OR) of k literals. A literal l is an occurrence of a Boolean variable or its complement. Hence, in order to satisfy a formula, each of its clauses must have at least one literal evaluated to true.

As an example, a CNF instance $f(a, b, c) = (a + \bar{b}) \cdot (a + b + c)$ consists of three variables, two clauses, and five literals. The assignment $\{a = 0, b = 1, c = 0\}$ leads to a conflict, whereas the assignment $\{a = 0, b = 0, c = 1\}$ satisfies f.

Despite the problem being NP-complete, there have been dramatic improvements in SAT solver technology over the past decade. This has led to the development of several powerful SAT solvers that are capable of handling problems consisting of thousands of variables and millions of constraints.[10–13]

Recently, SAT solvers[16,19–22] have been extended to handle PB constraints which are linear inequalities with integer coefficients that can be expressed in the normalized form[16] of

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b\,, \tag{1}$$

where $a_ib \in Z$ and x_i are Boolean variables. PB constraints can, in some cases, replace an exponential number of CNF constraints. They have been found to be very efficient in expressing "counting constraints".[16] Furthermore, PB extends SAT solvers to handle *optimization* problems as opposed to only *decision* problems. Subject to a given set of CNF and PB constraints, one can request the minimization (or maximization) of an objective function which consists of a linear combination of the problem's variables. Note that each CNF constraint can be viewed as a PB constraint. For example the CNF constraint $(\bar{a} + b)$ can be viewed as the PB constraint $\bar{a} + b \geq 1$. PB constraints represent 0-1 ILP inequalities.

2.3. *Symmetry detection and breaking for Boolean SAT*

Detecting and breaking symmetries in SAT instances have been shown to help prune the search space explored by a SAT solver. The basic framework for utilizing symmetries was proposed in Ref. 25, and later extended by Refs. 24 and 28 to account for phase-shift symmetries and consider only generators of the group of symmetries. In Ref. 1, the authors extended previous work to detect and break symmetries in SAT PB, i.e., 0-1 ILP, instances.

Symmetries in a SAT 0-1 ILP instance are first *detected* by reduction to graph automorphism and then *broken* by adding symmetry breaking predicates (SBPs) to the formulation. In the graph automorphism step, the instance is represented by a graph and the automorphism problem for that graph is solved using graph automorphism software packages, such as saucy.[29] SBPs, representing the generators of group of symmetries, are then added to the SAT instances in CNF clause format.

3. Problem Formulation and Implementation

In this section, we show how to formulate the state assignment problem as a 0-1 ILP instance. We used an approach similar to the one discussed in Ref. 1. The goal is to find the state assignment that leads to the minimum number of weighted transitions. To illustrate the approach, assume an FSM with x states and a defined weight between every two connected states. The number of bits, referred to as B, needed to encode each state is $\lceil \log_2 x \rceil$. The objective is to find a unique state assignment to each state while minimizing the weighted hamming distance between the adjacent states.

Two sets of variables are defined for the problem:

- A Boolean variable b_i^l that represents bit l of state S_i. A total of $x \log_2 x$ variables are defined. A value of 1 (0) for each variable indicates that the corresponding bit is a 1 (0) in the original problem.

- A Boolean variable $e^l_{i,j}$ that represents the XOR operation between the l bits of states S_i and S_j. A total of $\binom{x}{2}\log_2 x$ variables are defined. A value of 1 (0) for each variable indicates that the corresponding bit assignments are different (similar) in the original problem.

The following set of constraints is generated:

- The XOR relation between the e and b variables for all state bits must be defined using the following constraint:

$$\sum_{l=1}^{B}(e^l_{i,j} = b^l_i \oplus b^l_j) \quad \forall\, i, j \quad i \neq j\,. \tag{2}$$

Each XOR relation of the form $(r = s \oplus t)$ is expressed using four CNF constraints as follows:

$$(\bar{r} \vee s \vee t) \wedge (r \vee \bar{s} \vee t) \wedge (r \vee s \vee \bar{t}) \wedge (\bar{r} \vee \bar{s} \vee \bar{t})\,. \tag{3}$$

This relation yields a total of $4B\binom{x}{2}$ three-CNF constraints (i.e., each clause consists of the disjunction of three literals).

- Each state must have a unique state assignment. This is represented using the following PB constraint:

$$\sum_{l=1}^{B} e^l_{i,j} \geq 1 \quad \forall\, i, j, \quad i \neq j\,. \tag{4}$$

This relation yields a total of $B\binom{x}{2}$ PB constraints.

The optimization goal is to minimize the weighted hamming distance between the states. This is expressed using the following:

$$\text{Min}\left(\sum_{k=1}^{n} \omega^k_{i,j} \sum_{l=1}^{B} e^l_{i,j}\right) \quad \forall\, i, j, \quad i \neq j\,. \tag{5}$$

where n is the number of edges in the FSM and $w^k_{i,j}$ is the weight of the edge between states S_i and S_j.

3.1. *An illustrative example*

In this subsection, we use the FSM as shown in Fig. 1 to provide the reader with an example that clearly illustrates the various steps of the formulation. The shown FSM has a total of four states. Therefore, two bits are needed to represent each state. A maximum of $\binom{4}{2}$, i.e., six state combinations can exist. A total of 8b and 12e variables are declared. The XOR relation constraints are:

$$\begin{array}{llll} e^1_{A,B} = b^1_A \oplus b^1_B\,, & e^2_{A,B} = b^2_A \oplus b^2_B\,, & e^1_{B,C} = b^1_B \oplus b^1_C\,, & e^2_{B,C} = b^2_B \oplus b^2_C\,, \\ e^1_{C,D} = b^1_C \oplus b^1_D\,, & e^2_{C,D} = b^2_C \oplus b^2_D\,, & e^1_{B,D} = b^1_B \oplus b^1_D\,, & e^2_{B,D} = b^2_B \oplus b^2_D\,. \end{array} \tag{6}$$

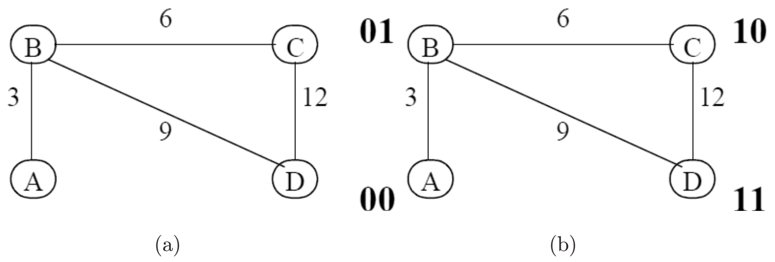

Fig. 1. (a) An illustrative example showing an FSM with four states and four edges. The weight of each edge is shown. (b) Shows the FSM with the corresponding state assignments (in bold).

Since only four edges exist, 8e variables were only used. The XOR relations generate a total of 32 three-CNF constraints.

The unique state assignment condition is expressed using the following four PB constraints:

$$e^1_{A,B} + e^2_{A,B} \geq 1\,, \quad e^1_{C,D} + e^2_{C,D} \geq 1\,, \quad e^1_{B,C} + e^2_{B,C} \geq 1\,, \quad e^1_{B,D} + e^2_{B,D} \geq 1\,. \tag{7}$$

Finally, the optimization goal is expressed using:

$$\text{Min}(3e^1_{A,B} + 3e^2_{A,B} + 6e^1_{B,C} + 6e^2_{B,C} + 12e^1_{C,D} + 12e^2_{C,D} + 9e^1_{B,D} + 9e^2_{B,D})\,. \tag{8}$$

The optimization instance is passed to the ILP solver which returns the assignment: $((b^1_A, b^2_A), (b^1_B, b^2_B), (b^1_C, b^2_C), (b^1_D, b^2_D)) = ((0,0)(0,1)(1,0)(1,1))$. The assignment yields the minimum possible optimization cost of 36 that the given FSM can experience. The cost is achieved as follows: 1 bit change between states A and $B(\text{cost} = 3) + 2$ bit changes between states B and $C(\text{cost} = 2 \times 6 = 12) + 1$ bit change between states C and $D(\text{cost} = 12), +$ and 1 bit change between states B and $D(\text{cost} = 9)$.

4. Experimental Results

In this section, we report and discuss the experimental results obtained for the state assignment technique. The results for the MCNC benchmark circuits[30] are presented in Table 1. We used the SAT-based 0-1 ILP solver MiniSAT+,[21] in addition to the generic commercial ILP solver, IBM-ILOG CPLEX 10.0.[23] All experiments were conducted on an Intel Xeon 3 GHz workstation running Linux and equipped with 4 GB of RAM. We used the default settings for MiniSAT+ and CPLEX. A time-out limit of 10,000 s was set for all experiments. Since some SAT solvers do not accept fractions for edge weights and to improve the results accuracy, weights were multiplied by 1M.

Table 1 lists the experimental results for MiniSAT+ and CPLEX. The first two columns show the name and size of the circuit; *Time* is the runtime in seconds needed to solve the problem; *Value* is the optimal cost achieved by the solver as explained in

Table 1. Experimental results using the SAT-based 0-1 ILP solver MiniSAT+ and the generic ILP solver CPLEX 10.0. Time is in seconds. Value represents the minimum optimization cost found by the solver.

Circuit		0 Extra bits				1 Extra bits				2 Extra bits			
		CPLEX		MiniSAT+		CPLEX		MiniSAT+		CPLEX		MiniSAT+	
Name	# States	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value
dk15	4	0.02	1060433	0	1060433	0.03	1060433	0.01	1060433	0.08	1060433	0.03	1060433
s8	5	0.13	161635	**0.05**	161635	0.49	161635	0.82	161635	0.99	161635	1.78	161635
s27	6	**0.65**	894772	3.9	894772	1.45	894772	14.82	894772	8.63	894772	14.48	894772
dkl4	7	**2.14**	1273863	17.58	1273863	29.34	1273863	122.91	1273863	60.05	1273863	369.3	1273863
dk27	7	**1.24**	1309519	2.38	1309519	10.8	1297614	7.28	1297614	60.2	1297614	82.43	1297614
dkl7	8	**2.9**	1063784	7.78	1063784	50.98	1063784	40.76	1063784	62.62	1063784	38.23	1063784
ex6	8	**3.98**	1071092	7.38	1071092	28.39	1071092	76.59	1071092	67.43	1064262	182.16	1064262
ex3	10	0.09	0	>10000	0	0.11	0	>10000	0	**0.06**	0	>10000	0
opus	10	22.59	835510	**11.28**	835510	355.8	835510	121.43	835510	2831	835510	482.6	835510
s386	13	569.2	967095	**42.57**	967095	>10000	910845	1216	910845	4583	899996	561.8	899996
ex4	14	126.2	478257	0.93	478257	30.36	478257	0.48	478257	24.52	478257	**0.32**	478257
dk512	15	>10000	1223202	**1229.2**	1218738	>10000	1200880	>10000	1200880	>10000	1200880	>10000	1200880
markl	15	>10000	1057205	863.5	1057205	>10000	1047797	**749.6**	1047797	>10000	1040749	1399	1040749
kirkm.	16	>10000	761685	**25.71**	761685	>10000	761608	149.4	761608	>10000	761540	167.9	761540
ex1	19	>10000	715921	>10000	723029	>10000	689671	>10000	712127	>10000	686277	>10000	1482254
ex2	19	1028.6	1000000	0.04	1000000	45.75	1000000	0.04	1000000	79.77	1000000	**0.03**	1000000
tma	20	>10000	250624	>10000	248733	>10000	249697	**5121**	248241	>10000	251161	>10000	248241
Total		51758	14124597	**32212**	14125350	60554	13997458	37622	14018458	57779	**13970733**	43301	14763790

*Number in bold represents the smallest figure in their respective rows.

Eq. (5). Since the solver does not accept fraction coefficients, all probability *Values* were multiplied by 10^6. The table also includes three categories labeled 0-, 1-, and 2-extra bits. In the first category, i.e., 0 extra bits, the minimum number of bits were used to represent the states, i.e., $\lceil \log_2 x \rceil$ where x is the number of states. Note that we are not constrained by using the minimum number of bits to encode the states, and hence we decided to experiment by increasing the number of bits and try different solutions. The increase in the number of bits needs to be reasonable otherwise this increase will lead to a complex combinational component of the circuit. The second and third categories, i.e., 1 and 2 extra bits, used $(\log_2 x + 1)$ and $(\log_2 x + 2)$ bits, respectively, to represent each state.

Overall, the results obtained show the superiority of MiniSAT+ over CPLEX in most instances. Although CPLEX outperformed MiniSAT+ in some smaller circuits, for complex larger circuits MiniSAT+ produced better results. For example, when considering the *kirkman* circuit, CPLEX timed-out after 10,000 s while MiniSAT+ found the optimal solution in 26 s.

Note that while both solvers did time-out on some instances, the solvers did identify some solutions almost instantly. The identified solutions are useful since they represent state assignments with a low number of weighted transitions. The identified solution could also be the optimal solution, except that the solver timed-out because it was busy trying to prove the solution's optimality. The Value column in Table 1 shows a value for all instances that did time-out. These values reflect the best identified solution within the 10,000 s of search. In Table 1, CPLEX times-out when solving the *mark*1 and *kirkm* instances while MiniSAT+ solves them in less than 900 s. However, the solution identified by CPLEX is in fact the optimal solution since its value is equivalent to the MiniSAT+ solution's value. The remaining instances that time-out with CPLEX, e.g., *dk*512, identify solutions with values close to the optimal solution's value identified by MiniSAT+. Perhaps, given some extra time CPLEX would have been able to identify the optimal solution.

When testing the advantage of adding extra bits to encode the states, results show that extra bits lead to better optimization cost values, but at the expense of increasing search runtimes and added circuit complexity.

Figure 2 shows the FSM of the *s*8 instance and its corresponding state assignment. The circuit has five states and hence, needs three bits to represent each state. The minimal optimization cost which represents the minimum weighted transitions is 161635.

In an effort to prune the search space and speed up the ILP solver, the tested instances were preprocessed and checked for symmetries, using the ShatterPB tool.[31] As shown in Table 2, ShatterPB was able to find symmetries in all instances in a short amount of time. Table 3 shows the results after detecting the symmetries and adding SBPs to the instances. Upon comparing the results of Tables 1 and 3, it is evident that MiniSAT+ significantly benefited from the addition of the SBPs. For

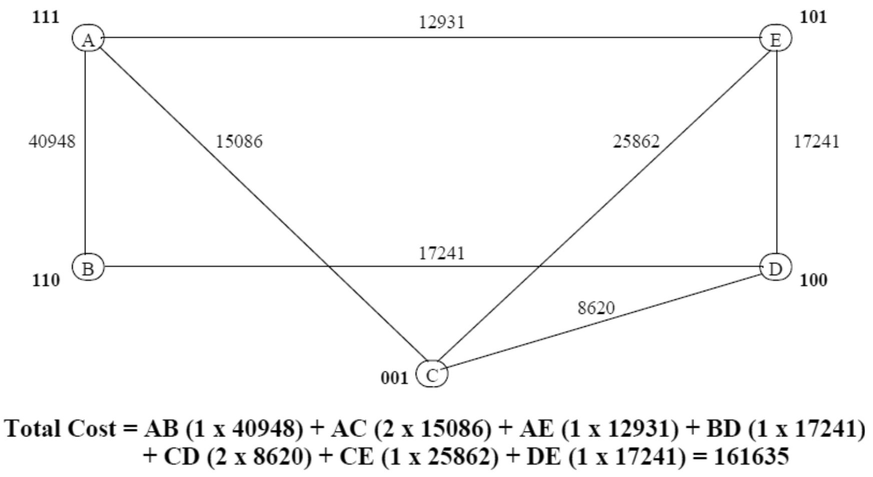
111
A
12931
101
E
40948
15086
25862
17241
110
B
17241
D
100
8620
001
C
Total Cost = AB (1 x 40948) + AC (2 x 15086) + AE (1 x 12931) + BD (1 x 17241) + CD (2 x 8620) + CE (1 x 25862) + DE (1 x 17241) = 161635

Fig. 2. An illustrative example showing the FSM and the corresponding state assignments for the s8 instance. The instance has a total of 5 states and needs 3 bits to represent each state. The minimum possible optimization cost is 161635.

Table 2. Symmetry breaking results. S, G, and T represent the number of symmetries, number of generators, and the time (in seconds) needed by ShatterPB to find the symmetries, respectively.

Circuit		0 Extra bits			1 Extra bits			2 Extra bits		
Name	# States	S	G	T	S	G	T	S	G	T
dkl5	4	2	1	0.01	6	2	0	24	3	0
s8	5	6	2	0	24	3	0.01	120	4	0
s27	6	6	2	0	24	3	0	120	4	0.01
dkl4	7	6	2	0.01	24	3	0.01	120	4	0.01
dk27	7	6	2	0.01	24	3	0	120	4	0.01
dkl7	8	6	2	0.01	24	3	0.02	120	4	0.03
ex6	8	6	2	0.01	24	3	0.02	120	4	0.03
ex3	10	8.7E7	12	0.22	4.4E8	13	0.33	2.6E9	14	0.5
opus	10	24	3	0.04	120	4	0.07	720	5	0.1
s386	13	24	3	0.11	120	4	0.19	720	5	0.29
ex4	14	24	3	0.14	120	4	0.24	720	5	0.38
dk512	15	24	3	0.19	120	4	0.33	720	5	0.5
mark1	15	48	4	0.24	240	5	0.41	1440	6	0.63
Kirk.	16	24	3	0.25	120	4	0.41	720	5	0.65
ex1	19	120	4	0.8	120	4	1.06	24	3	1.04
ex2	19	5040	6	1.04	120	4	1.02	24	3	1.05
tma	20	120	4	1	24	3	1.03	6	2	1.07

example, in the *tma* circuit, MiniSAT+ was able to solve the instance with symmetries in 8524 s while it timed-out when solving the original instance. In terms of the effect of symmetry breaking on the performance of CPLEX, the results are mixed. For the category with 0 extra bits, CPLEX performed better on two instances

Table 3. Experimental results using the SAT-based 0-1 ILP solver MiniSAT+ and the generic ILP solver CPLEX 10.0. All instances were preprocessed with ShatterPB and updated with symmetry breaking predicates (SBPs). Time (in seconds) represents the solver runtime. Value represents the minimum optimization cost found by the solver.

Circuit		0 Extra bits w/SBP				1 Extra bits w/SBP				2 Extra bits w/SBP			
		CPLEX		MiniSAT+		CPLEX		MiniSAT+		CPLEX		MiniSAT+	
Name	# States	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value
dk15	4	0.02	1060433	**0**	1060433	0.06	1060433	0.01	1060433	0.22	1060433	0.01	1060433
s8	5	0.21	161635	**0.03**	161635	0.62	161635	0.6	161635	1.75	161635	1.03	161635
s27	6	**0.6**	894772	2.03	894772	2.77	894772	4.27	894772	10.46	894772	8	894772
dkl4	7	**2.53**	1273863	7.25	1273863	15.21	1273863	14.4	1273863	76.99	1273863	88.25	1273863
dk27	7	2.07	1309519	**1.45**	1309519	12.88	1297614	2.34	1297614	42.68	1297614	4.91	1297614
dkl7	8	5.58	1063784	**4.43**	1063784	16.85	1063784	9.76	1063784	61.19	1063784	32.91	1063784
ex6	8	4.9	1071092	**3.6**	1071092	22.58	1071092	9.92	1071092	357.4	1064262	41.2	1064262
ex3	10	**0.45**	0	>10000	0	0.76	0	>10000	0	0.84	0	>10000	0
opus	10	50.09	835510	**3.35**	835510	145.4	835510	7.37	835510	1294	835510	12.28	835510
s386	13	952.6	967095	**11.32**	967095	4122	910845	61.67	910845	7043	899996	42.18	899996
ex4	14	139	478257	0.94	478257	44.64	478257	**0.7**	478257	79.29	478257	1.44	478257
dk512	15	>10000	1224690	**144**	1218738	>10000	1200880	1015.5	1200880	>10000	1200880	2109	1200880
markl	15	7438	1057205	**120.4**	1057205	>10000	1047797	353.45	1047797	>10000	1041977	320.3	1040749
kirkm.	16	>10000	761685	**5.14**	761685	>10000	761608	19.53	761608	>10000	761540	27.32	761540
ex1	19	>10000	703609	>10000	681296	>10000	699856	>10000	680282	>10000	694910	>10000	726522
ex2	19	57.07	1000000	0.07	1000000	96.75	1000000	0.06	1000000	83.72	1000000	**0.05**	1000000
tma	20	>10000	253692	8524	248733	>10000	252887	**7463**	248241	>10000	249029	>10000	249469
Total		48653	14116841	**28828**	14083617	54480	14010833	28963	13986613	59052	**13978462**	32689	14009286

*Number in bold represents the smallest figure in their respective rows.

(*mark*1 and *ex*2), but worse on six instances. This observation agrees with earlier work[26] that suggests that the generic ILP solver CPLEX is actually slowed down by the addition of SBPs. Since CPLEX is a commercial tool and the algorithms used by it are not publicly known, it is difficult to pinpoint a reason for this disparity.

5. Conclusion

Starting from a probabilistic description of a FSM, this work has attempted to find a state assignment solution that minimizes the switching activity of the state variables. The main contributions of this work are as follows: (1) We showed how to formulate the assignment problem as a SAT 0-1 ILP problem. (2) We experimented and compared the performance of advanced Boolean SAT and generic ILP solvers when solving the SAT 0-1 ILP state assignment problem. Results indicate that for larger topologies, the SAT-based 0-1 ILP solver, MiniSAT+, outperforms the commercial generic ILP solver, CPLEX. (3) We showed empirically, for the presented sample of MCNC instances, that increasing the number of state variables does not lead to noticeable improvement in reaching an optimal solution. (4) We tested the state assignment instances for the existence of symmetries and were able to detect symmetries in all instances. The detected symmetries were broken and used to improve the search runtime. Presented results indicate that detecting and breaking symmetries in such instances significantly improves the 0-1 ILP SAT solver runtime. As far as we know, this is the first work that shows that state assignment problem contains symmetries that can be utilized to speed up search. The effects of the code assigned using the technique presented here on the switching activity of the combinational part of the circuits will be studied in future work.

Acknowledgment

This research was partially supported by the European Union through the European Regional Development Fund.

References

1. L. Benini and G. De Micheli, State assignment for low power dissipation, *IEEE J. Solid-State Circuits* **30** (1995) 258–268.
2. G. Hachtel, M. Hermida, A. Pardo, M. Poncino and F. Somenzi, Re-encoding sequential circuits to reduce power dissipation, *Proc. Int. Conf. Computer Aided Design* (1994), pp. 70–73.
3. M. Koegst, G. Franke and K. Feske, State assignment for FSM low power design, *Proc. European Design Automation Conf.* (1996), pp. 28–33.
4. L. Daldoss, D. Sciuto and C. Silvano, State encoding for low power embedded controllers, *Proc. IEEE Int. Symp. Circuits and Systems* (1998), pp. 421–424.
5. W. Noth and R. Kolla, Spanning tree based state encoding for low power dissipation, *Proc. Design Automation and Test in Europe Conf.* (1999), pp. 168–174.

6. P. Bacchetta, L. Daldoss, D. Sciuto and C. Silvano, Low-power state assignment techniques for finite state machines, *Proc. IEEE Int. Symp. Circuits and Systems* (2000), pp. 641–644.
7. R. Eggermont, S. Cotofana and C. Lageweg, Profiling-based state assignment for low power dissipation, *Proc. Program for Research on Integrated Systems and Circuits* (2004).
8. S. Park, S. Cho, S. Yang and M. Ciesielski, A new state assignment technique for testing and low power, *Proc. Design Automation Conf.* (2004), pp. 510–513.
9. L. Yuan, G. Qu, T. Villa and A. Sangiovanni-Vincentelli, FSM Re-engineering and its application in low power state encoding, *Proc. Asia and South Pacific Design Automation Conf.* (2005), pp. 254–259.
10. G. Audemard and L. Simon, Predicting learnt clauses quality in modern SAT solver, *Proc. Int. Joint Conf. Artificial Intelligence* (2009), pp. 399–404.
11. A. Biere, PicoSAT essentials, *J. Satisfiability, Boolean Model. Comput.* **4** (2008) 75–97.
12. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik, Chaff: Engineering an efficient SAT solver, *Proc. Design Automation Conf.* (2001), pp. 530–535.
13. K. Pipatsrisawat and A. Darwiche, RSat 2.0: SAT solver description, Technical report D153, Automated Reasoning Group, Computer Science Department, University of California, Los Angeles (2007).
14. A. Biere, A. Cimatti, E. Clarke, M. Fujita and Y. Zhu, Symbolic Model Checking using SAT procedures instead of BDDs, *Proc. Design Automation Conf.* (1999), pp. 317–320.
15. G. Nam, F. Aloul, K. Sakallah and R. Rutenbar, A comparative study of two Boolean formulations of FPGA detailed routing constraints, *IEEE Trans. Comput.* **53** (2004) 688–696.
16. F. Aloul, A. Ramani, I. Markov and K. Sakallah, Generic ILP versus specialized 0-1 ILP: An update, *Proc. Int. Conf. Computer-Aided Design* (2002), pp. 450–457.
17. S. Memik and F. Fallah, Accelerated Boolean satisfiability-based scheduling of control/data flow graphs for high-level synthesis, *Proc. Int. Conf. Computer Design* (2002), pp. 395–400.
18. A. Sagahyroon and F. Aloul, Using SAT-based techniques in power estimation, *Microelectron J.* **38** (2007) 706–715.
19. D. Chai and A. Kuehlmann, A fast pseudo-Boolean constraint solver, *Proc. Design Automation Conf.* (2003), pp. 830–835.
20. H. Dixon and M. Ginsberg, Inference methods for a pseudo-Boolean satisfiability solver, *Proc. National Conf. Artificial Intelligence* (2002), pp. 635–640.
21. N. Een and N. Sorensson, An extensible SAT-solver, *Proc. Int. Conf. Theory and Applications of Satisfiability Testing* (2003), pp. 502–508.
22. H. Sheini and K. Sakallah, Pueblo: A modern pseudo-Boolean SAT solver, *Proc. Design, Automation, and Test Conf. Europe* (2005), pp. 684–685.
23. ILOG CPLEX, http://www.ilog.com/products/cplex.
24. F. Aloul, A. Ramani, I. Markov and K. Sakallah, Solving difficult instances of Boolean satisfiability in the presence of symmetry, *IEEE Trans. Comput.-Aided Des.* **22** (2003) 1117–1137.
25. J. Crawford, M. Ginsberg, E. Luks and A. Roy, Symmetry-breaking predicates for search problems, *Proc. 5th Int. Conf. Principles of Knowledge Representation and Reasoning* (1996), pp. 148–159.
26. A. Ramani, F. Aloul, I. Markov and K. Sakallah, Breaking instance-independent symmetries in exact graph coloring, *J. Artif. Intell. Res.* **26** (2006) 289–322.

27. J. Nelson, K. Tumer and J. Ghosh, Designing genetic algorithms for the state assignment problem, *IEEE Trans. Syst. Man Cybern.* **25** (1999) 687–694.
28. F. Aloul, K. Sakallah and I. Markov, Efficient symmetry breaking for Boolean satisfiability, *IEEE Trans. Comput.* **55** (2006) 549–558.
29. P. Darga, M. Lifiton, K. Sakallah and I. L. Markov, Exploiting structure in symmetry generation for CNF, *Proc. Design Automation Conf.* (2004), pp. 530–534.
30. MCNC Benchmarks, http://www.cbl.ncsu.edu/CBL_Docs/Bench.htm.
31. F. Aloul, A. Ramani, I. Markov and K. Sakallah, Symmetry-breaking for pseudo-Boolean formulas, *ACM J. Exp. Algorithmics* **12** (2007).