Computing Subsets of Equivalence Classes for Large FSMs

Gianpiero Cabodi Stefano Quer

Dip. di Automatica e Informatica
Politecnico di Torino
Turin, Italy

Paolo Camurati

Dip. di Matematica e Informatica
Università di Udine
Udine, Italy

Abstract

Computing equivalence classes for FSMs has several applications to synthesis and verification problems. Symbolic traversal techniques are applicable to medium-small circuits. This paper extends their use to large FSMs by means of cofactor-based enhancements to the state-of-the-art approaches and of underestimations of equivalence classes. The key to success is pruning the search space by constraining it. Experimental results on some of the larger ISCAS'89 and MCNC circuits show its applicability.

1 Introduction

Finding the classes of equivalent states of a Finite State Machine (FSM) has several applications to automated synthesis and to formal verification [1], [2]. Devices obtained by composing interacting FSMs typically contain many equivalent states. State minimization can lead to more efficient implementations with fewer latches. For verification, it is important to extract a reduced machine, also referred to as the quotient machine, to prove properties, e.g., model checking or language containment, in a more efficient way.

The problem of finding equivalence classes is well-known from graph theory, but the algorithms for solving it are exponential in the number of latches and require the explicit representation of the state transition graph. For this reason they are applicable just to very small circuits.

Binary Decision Diagrams (BDDs) and symbolic traversal techniques for FSMs represented a major breakthrough in terms of efficiency and applicability. BDDs represent both Boolean functions and sets of states, supporting also their manipulation by means of logical operators. Symbolic traversal techniques explore the state space of the FSM, either in forward or backward breadth-first mode, their goal being to prove global properties on a step by step basis.

The basic algorithm for identifying classes of equivalent states with symbolic traversals is due to Lin *et al.* [1]. They are able to compute the equivalence classes of the smaller ISCAS'89 and MCNC benchmarks.

An expensive step in Lin's algorithm is computing inverse images in the state space of the product machine. Tamisier [2] uses a variant of the *compatible projector* called *cross section* to simplify inverse image computation. The gain in terms of BDD size and CPU time is considerable, yet it doesn't allow to deal with bigger circuits. In fact, even if the intermediate steps are simpler, the BDDs for the state sets explode with large FSMs.

The inability to represent large state sets occurred also in symbolic traversals and fostered the development of approximate techniques. For example, Cho *et al.* [3] perform an approximate forward symbolic traversal to explore the state space of large machines, computing an over-estimation of the reachable state set. Over-estimations are used by Cabodi *et al.* [4] to prune the search space during backward traversal when trying to verify the input/output equivalence of two machines.

Approximate solutions are a way to attack large circuits. We compute underestimations, i.e., subsets, of the equivalence classes. We divide the search space in smaller subspaces and we compute subsets of the equivalence relation. Eventually, several underestimations are properly combined to reach a good approximation of the global equivalence relation. This reduces the amount of information to be manipulated, extending the applicability of these techniques to larger circuits.

The main contributions of this paper are:

- we improve on the basic techniques of [1] and [2] by extensively using cofactoring for function and state set simplification during traversals
- we define the notion of underestimation of an equivalence classes, by restricting the working spaces of the product machine with proper constraints.
- we show how to combine approximate results to obtain a better approximation
- we present for the first time experimental results on the equivalent states for some of the larger ISCAS'89 and MCNC benchmark circuits.

The remainder of the paper is organized as follows. In the next section, we introduce the basic definitions. Section 3 summarizes the approaches of Lin *et al.* and Tamisier. Section 4 presents our cofactor-based improvements to the original tecniques. Section 5 introduces the fundamental notions about underestimating an equivalence class and the related algorithms. In Section 6

experimental evidence shows that we are able to deal with some of the larger benchmark circuits known from the literature. Section 6 closes the paper with a brief summary, outlining future developments.

2 Preliminaries

2.1 Finite state machines

A *finite state machine* is an abstract model describing the behavior of a sequential circuit. A completely specified FSM M is a 5-tuple $M' = (I, O, S, \delta, \lambda)$, where I is the input alphabet, O is the output alphabet, S' is the state space, δ' is the next state function ($\delta' : S' \times I \rightarrow S'$), and λ' is the output function ($\lambda' : S' \times I \rightarrow O$).

The *product machine* of two FSMs M' and M'' is a machine $M = M' \times M'' = (I, \{0,1\}, S, \delta, \lambda)$, where $s' = (s'_1, \ldots, s'_n)$ and $s'' = (s''_1, \ldots, s''_n)$ are the state variables of the component machines, $S = S' \times S''$ is the state space, $\delta(s', s'', x) = (\delta'(s', x), \delta''(s'', x))$ is the next state function, and $\lambda(s', s'', x) = (\lambda'(s', x) \equiv \lambda''(s'', x))$ is the output function. We shall hereinafter consider product machines composed of two copies M' and M'' of the same FSM.

2.2 Image and inverse image

Let $f : \mathcal{B}^i \rightarrow \mathcal{B}^j$ be a Boolean function and $C \subseteq \mathcal{B}^i$ a subset of its domain. The image of C according to f is:

$$\textsc{Img}(f, C) = \{y \in \mathcal{B}^j \mid \exists x \in C \ \wedge \ y = f(x)\}$$

Subset C is often called *constraint*.

Let $f : \mathcal{B}^i \rightarrow \mathcal{B}^j$ be a Boolean function and $C \subseteq \mathcal{B}^j$ be a subset of its range. The inverse image of C according to f is:

$$f^{-1}(C) = \{x \in \mathcal{B}^i \mid \exists y \in C \ \wedge \ y = f(x)\}$$

2.3 Equivalent states and classes

Two states $s', s'' \in S$ are equivalent iff their outputs are the same for all possible inputs and their image according to δ consists of equivalent states.

From the notion of equivalent states one can define the *equivalence relation* [5] $E \subseteq S' \times S'' : E = \{(s', s'') \mid s' \approx s''\}$ satisfying the following properties:

- reflexivity: $\forall s' \in S' \Rightarrow (s', s') \in E$
- symmetry: $(s', s'') \in E \Rightarrow (s'', s') \in E$
- transitivity: $(s', s'') \in E \wedge (s'', s''') \in E \Rightarrow (s', s''') \in E$.

Following the notation of Lin *et al.* [1], the equivalence class of s' is denoted as $[s'] = \{s'' \mid (s', s'') \in E\}$. As the number of equivalence classes can be very large, the algorithms of the next section aim at finding the characteristic function $E(s', s'')$ of the set that contains all equivalent states.

3 State-of-the-art equivalence class computation

Given the definition of equivalent states, Lin *et al.* [1] suggested a fixed point computation that returns the characteristic function $E(s', s'')$ of the set of all equivalent states. The starting point is the set of states for which, for all possible input values, the output functions don't differ ($E_0(s', s'') = \forall x \, \lambda(s', s'', x)$). These states are *1-equivalent*, according to Lin's terminology. The formula gives the set of *k+1-equivalent* states as a logical conjunction of two sets:

- states whose image consists, for all inputs, of *k-equivalent* states ($\forall x \, \delta^{-1}(E_k(s', s''))$)
- *1-equivalent* states, i.e., belonging to $E_0(s', s'')$

that is:

$$\begin{array}{rcl} E_0(s', s'') & = & \forall x \, \lambda(s', s'', x) \\ E_{k+1}(s', s'') & = & E_0 \cdot \forall x \, \delta^{-1}(E_k) \end{array} \qquad (1)$$

Note that if $A \subseteq B$, then $f^{-1}(A) \subseteq f^{-1}(B)$. As a consequence, at each iteration $E_{k+1}(s', s'') \subseteq E_k(s', s'')$. We modify formula (1) performing a logical conjunction with the *k-equivalent* states ($E_k(s', s'')$) instead of the *1-equivalent* ones:

$$E_{k+1}(s', s'') = E_k \cdot \forall x \, \delta^{-1}(E_k) \qquad (2)$$

In a dual way, noting that $E(s', s'') = \overline{D(s', s'')}$, where $D(s', s'')$ is the characteristic function of the set of non equivalent states, the fixed point computation starts from the set of *1-distinguishable* states:

$$\begin{array}{rcl} D_0(s', s'') & = & \exists x \, \overline{\lambda(s', s'', x)} \\ D_{k+1}(s', s'') & = & D_k + \exists x \, \delta^{-1}(D_k) \end{array}$$

Let $E(s', s'') \subseteq S' \times S''$ be a Boolean equivalence relation and let $r \in S'$ be a reference state. The *compatible projection* of E with respect to r is the function Cproj defined as follows [5]:

$$\textsf{Cproj}(E, r) = \{(s', s'') \mid (s', s'') \in E \wedge s'' = \textsc{sel}(s')\}$$

where SEL is a function that uniquely selects a member of the equivalence class $[s']$ of s'. The criterion is based on minimum distance from the reference state r [5]. Note that we consider a unique reference point for computing distances while applying the *compatible projector*.

Lin *et al.* discovered that many state pairs are *single cycle equivalent*, i.e., the next states and the outputs are the same for all possible inputs. A simple combinational operation results in an underestimation of the equivalence classes:

$$E^-(s', s'') = \forall x((\delta'(s', x) \equiv \delta''(s'', x)) \cdot \lambda((s', s''), x))$$

Once $E(s', s'')$ is known, Lin and Newton [5] use it to compute a state-minimal transition relation of the FSM. For this purpose they introduce the *compatible projector*.

Its intuitive goal is to select just one member for each equivalence class, based on the notion of minimal distance from a reference state, i.e., to transform relation $E(s', s'')$ into a compatible function.

Tamisier [2] believes that most of the cost is due to computing inverse images in the state space of the product machine. His approach exploits a variant of the compatible projector, called *cross section* to constrain the state transition function $\delta(s', s'', x)$. Composing the *cross section* $\mathcal{C}$ with the state transition function δ on the set of *k-equivalent* states yields a modified function δ^*:

$$\delta^* = \mathcal{C} \circ \delta'$$

The computation of δ^* includes an inverse image computation step that is simpler, because it is within the state space of a single machine. This justifies the experimental advantages reported by Tamisier in [2] both in terms of BDD nodes and of CPU time.

Eventually Tamisier proves that, for all inputs, the inverse image of the set of *k-equivalent* states according to $\delta(s', s'', x)$ coincides with a Boolean equality of two copies of δ^* working on $s' \in S'$ and $s'' \in S''$ variables:

$$\forall x \, \delta^{-1}(E_k(s', s'')) = \forall x \, (\delta^*(s', x) \equiv \delta^*(s'', x))$$

Nevertheless, this method is not applicable to larger FSMs. We point out the following reasons for failure:

- Tamisier doesn't completely work in the state space of a single machine, because the last step is performed in the state space of the product machine
- inverse image computation is simpler, but the E relations that serve as input and outputs are not simpler, rather their complexity grows with the size of the FSM.

Our approach works in the state space of the product machine, but our inverse image computation is simpler, because of the use of constraints to prune the state space and because of more efficient function composition particularly suited to handling product machines.

4 Enhanced algorithms

Our enhancement to Lin's *et al.* method consists of two optimizations. We present them for the computation of equivalent states, but similar enhancements can be applied also in the case of distinguishable states. The logical conjunction with $E_k(s', s'')$ means that outside this set $E_{k+1}(s', s'') = 0$. By definition of generalized cofactor $\downarrow$, $A \cdot B = A \cdot B \downarrow A$. As a consequence, formula (2) becomes:

$$E_{k+1} = E_k \cdot (\forall x \, \delta^{-1}(E_k)) \downarrow E_k \qquad (3)$$

Cabodi *et al.* showed in that, if the pre-image of a set of states C is restricted by cofactoring with a constraint A, then A can be pulled inside to simplify also the δ functions:

$$\delta^{-1}(C) \downarrow A = (\delta \downarrow A)^{-1}(C)$$

As a consequence, acting $E_k(s', s'')$ as a constraint, formula (3) becomes:

$$E_{k+1} = E_k \cdot (\forall x \, (\delta \downarrow E_k)^{-1}(E_k)) \qquad (4)$$

Cabodi *et al.* also showed in that, if the pre-image of a set of states C is restricted by cofactoring with a constraint A, then one can simplify set C by cofactoring with B, an over-estimation of the image of the constraint:

$$\delta^{-1}(C) \downarrow A = \delta^{-1}(C \downarrow B) \downarrow A$$

where $B \supseteq \delta(A)$. According to formula (2), $E_k \subseteq E_{k-1}$ and $E_{k-1} \supseteq \delta(E_k)$. As a consequence, formula (3) becomes:

$$E_{k+1} = E_k \cdot (\forall x \, \delta^{-1}(E_k \downarrow E_{k-1})) \downarrow E_k \qquad (5)$$

Combining formulas (4) and (5), we eventually obtain

$$E_{k+1} = E_k \cdot (\forall x \, ((\delta \downarrow E_k)^{-1}(E_k \downarrow E_{k-1}))) \qquad (6)$$

5 Underestimating equivalence classes

The methods described in the previous section are applicable just to medium-small FSMs.

Some of the reasons are the number of variables describing the state space and the size of the BDDs, particularly important when the FSM is a product machine. Our approach to this problem is to consider, instead of the whole space, just a subset of it, defined by the characteristic function of a constraint. Reducing the space makes computations easier, at the expense of completeness, because only an underestimation of the equivalence classes will be found. We shall show in the next paragraphs how to reduce the state space by imposing a constraint, how to combine several underestimations and how to select constraints.

5.1 Underestimating $E(s', s'')$

Let $C(s', s'')$ be the characteristic function of a constraint on S. $C(s', s'')$ defines a subset of S:

$$C(s', s'') \subseteq S$$

Independently of how the constraint is chosen, once it is given, it is straightforward to modify the fixed point computation of equation (2) to compute a subset $E^C(s', s'') \subseteq E(s', s'')$:

$$\begin{aligned} E^C_0(s', s'') &= E_0 \cdot C \\ E^C_{k+1}(s', s'') &= E^C_k \cdot \forall x \, \delta^{-1}(E^C_k) \end{aligned} \qquad (7)$$

The enhancements of formula (6) are easy to implement in (7).

Once the fixed point is reached, it is possible to prove that:

$$E^C(s', s'') \subseteq E(s', s'') \cdot C(s', s'')$$

5.2 Combining underestimations

Restricting the analysis to a single constraint $C(s', s'')$ may yield a poor underestimation $E^C(s', s'')$. A way to obtain a result closer to the exact one is to consider several constraints, computing several underestimations, and then combining them together.

Let $C_1(s', s''), C_2(s', s''), \ldots, C_q(s', s'') \subseteq S$ be q symmetric, reflexive, and transitive subsets of the S state space. Subsets can in general overlap.

For each subset C_j we compute $E^{C_j}(s', s'')$ according equation (7). In order to find an appropriate underestimation E^-, we could combine them according to the following theorem by taking the transitive closure of the union of the subsets:

Theorem 1:
$E^-(s', s'') = \mathsf{T\_closure}(\cup_{j=1}^{q} E^{C_j}(s', s'')) \subseteq E(s', s'')$.

The goodness of the result E^-, i.e., how close it is to E, depends on the coupling of the subsets. If they are loosely coupled, computing the transitive closure will yield little benefit. Otherwise, it can significantly increase accuracy.

Although the individual underestimations $E^{C_j}(s', s'')$ are reasonably simple, computing their transitive closure may be a hard task. We must trade-off computational efficiency for better approximations. We look for a more approximate underestimation E^* provided it is a subset of E^-. As the ultimate goal is to apply the *compatible projector* to select a single state out of each equivalence class, instead of computing first the transitive closure and then its compatible projection, we prefer to immediately compute the compatible projections of the subsets and then compose them.

Theorem 2:
$\exists E^* \subseteq E^- \subseteq E \mid \mathsf{Cproj}(E^*) = \mathsf{Cproj}(E^{C_1}) \circ \mathsf{Cproj}(E^{C_2}) \circ \ldots \circ \mathsf{Cproj}(E^{C_q})$.

Composition is defined as follows:

$$(A \circ B)(s', s'') = \exists \sigma (A(s', \sigma) \cdot B(\sigma, s''))$$

The computational advantage due to theorem 2 is evident, because the composition of projections avoids the computation of the transitive closure.

5.3 Selecting the constraint

As the equivalence relation is symmetric, reflexive, and transitive, among the possible constraints we select those that share the same properties. To make the constraint really helpful, it is important that it shrinks the size of the BDDs. We reach this objective by reducing the number of state variables appearing in the constraint, imposing a relationship between some of the s' variables and the corresponding ones in s''.

Let $S' = S'' = \mathcal{B}^n$ and $S = S' \times S'' = \mathcal{B}^{2n}$, let τ be the set of the integers ranging from 1 to n: $\tau = \{1, \ldots, n\}$, and let α be a subset of τ. We use α to select among the literals of s the ones whose subscripts belong to α:

$$\begin{array}{rcl} s'_\alpha & = & \{s'_i \mid i \in \alpha\} \\ s''_\alpha & = & \{s''_i \mid i \in \alpha\} \\ s_\alpha & = & s'_\alpha \cup s''_\alpha \end{array}$$

Among the many relationships, we impose that some variable pairs are always equal, whereas the remaining ones may differ. In this case the constraint is the characteristic function of the subset of S such that the variable pairs not belonging to s_α are equal:

$$C_\alpha(s', s'') = \Pi_{i \in \tau - \alpha}(s'_i \equiv s''_i)$$

With this choice, we will handle equivalent states differing only in the s_α variables. The dimension of the state space is correspondingly reduced from 2^{2n} to $2^{n+\mathsf{Card}(\alpha)}$, where $\mathsf{Card}\ (\alpha)$ is the cardinality of set α. The size of the BDDs experiences a considerable reduction, too. There is another simplification that allows us to make the BDDs representing the equivalence classes smaller. At any equivalence class computation step, given a subset β of the literals, we retain only the pairs of equivalent states whose configuration in β doesn't depend on variables outside β: $E^\beta = \forall s_{\tau-\beta} E$. Experience shows that a wise choice for β is $\beta \supseteq \alpha$. As a consequence, the worst-case complexity of the BDDs is reduced from $O(2^{2n})$ to $O(2^{\mathsf{Card}(\alpha)+\mathsf{Card}(\beta)})$. By properly choosing α and β we can make our computations almost independent of the size n of the state space.

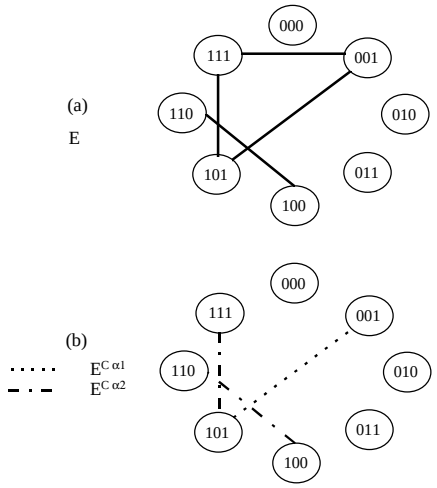

Figure 1: Exact equivalence classes and their underestimations for s27: part 1

5.4 Example

Let us apply the concepts of the previous paragraphs to compute an underestimation of the equivalence classes

of s27, the smallest ISCAS'89 benchmark. This circuit has 4 inputs, 3 latches, and 1 output.

Assuming a proper state variable ordering, and a $s_1 s_2 s_3$ encoding, Fig. 1(a) shows the equivalence relation graph E. Edges connect equivalent states, and self equivalence is omitted for simplicity, i.e., self loops are not indicated.

Equivalence classes with cardinality greater than 1 are $\{001, 101, 111\}$ and $\{100, 110\}$. Let us restrict the space as follows:

- $\alpha_1 = \{1\}$, $C_{\alpha_1}(s', s'') = (s'_2 \equiv s''_2) \cdot (s'_3 \equiv s''_3)$ (set of states of the product machine where only the most significant bit may differ)
- $\alpha_2 = \{2, 3\}$, $C_{\alpha_2}(s', s'') = (s'_1 \equiv s''_1)$ (set of states of the product machine where bits other than the most significant bit may differ).

The corresponding restricted equivalence classes ($E^{C_{\alpha_1}}$, $E^{C_{\alpha_2}}$) are shown in Fig. 1(b). It is easy to observe that, as stated by theorem 1, merging the two relations and taking the transitive closure of the resulting graph leads exactly to the global equivalence relation of Fig. 1(a).

Fig. 2(a) and Fig. 2(b) illustrate, respectively, the compatible projections of relations E, $E^{C_{\alpha 1}}$, and $E^{C_{\alpha 2}}$. Directed edges are used to represent remapping of states onto selected ones. The all-0 state serves as reference state r. There is an implicit self-loop for any node with no outgoing edge to itself.

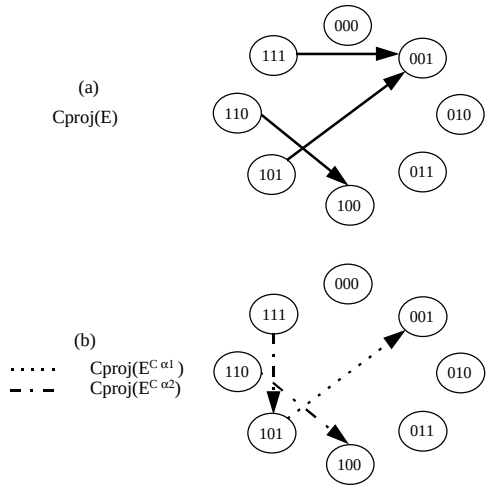

Figure 2: Exact equivalence classes and their underestimations for s27: part 2

6 Experimental results

The procedure for generating the equivalence classes has been implemented in a fully home-made package amounting to about 15,000 lines of C-code. BDD nodes are limited to 1,500,000. Experiments ran on a 130 MIPS DEC Alpha.

Tab. 1 collects some data about some ISCAS'89 and MCNC benchmarks. For each circuit it shows the number of primary inputs, primary outputs, latches, and gates.

Tab. 2 compares our enhanced method (EM) to the original method (OM) by Lin *et al.* For each circuit it shows the number of backward steps performed, the number of equivalence classes # EqC, including single state ones, and the CPU times without and with the improvements of section 3. The gain in term of time is evident. We obtain similar results in term of memory occupation although we don't present evidence on that. These results show that our enhanced method is at least comparable with Tamisier's [2] method that claims a gain in time by a factor ranging from 1.5 and 12 and in memory by a factor from 1.5 and 3.

Tab. 3 compares the approximate method resulting from single cycle equivalence (SCEM) [1] with the best case of the space reduction method (SRM) for circuits of Tab. 2 requiring more than 1.0s CPU time and for some larger circuits. # N is the number of underestimation we compute and combine in each case. Single cycle equivalence yields rather poor results, i.e., not close to the exact ones, and suffers from the exponentiality of the state space. The time required by single cycle equivalence check is reasonably low for small circuits, but for the larger ones, like s1423, s5378, $\mathsf{m\_m}_9$ (min-max MCNC circuit), the method is not applicable. It is easy to conclude that we can obtain a good level of approximation in the smaller circuits and, unlike exact techniques, we can also deal with some of the larger circuits (s1423 and s5378).

7 Conclusions

Computing equivalence classes has several applications.

Algorithms based on symbolic state space traversals already represent a notable achievement, but they are unable to deal with other than medium-small FSMs. There are two main contributions of this paper. First, an enhancement to state-of-the-art exact techniques based on state space pruning and cofactoring. Second, the notion of underestimation of equivalence classes. This is done by restricting the search space of the product machine by means of constraints.

Future work will consist in applying this approach to other circuits and in deepening our knowledge of state space reduction, e.g., making it automatic and developing topology-based heuristics. Accuracy increases when the subsets are strongly coupled, thus quite large. Efficiency requires smaller subsets to have smaller BDDs. We will also investigate heuristic tuning strategies to trade-off between accuracy and efficiency while selecting the subsets.

References

[1] B. Lin, H.J. Touati, A. Richard Newton, "Don't Care Minimization of Multi-Level Sequential Logic Networks," in *Proc. IEEE ICCAD'90*, November 1990, pp. 414–417

[2] T. Tamisier, "Computing the Observable Equivalence Relation of a Finite State Machine," in *Proc. IEEE ICCAD'90*, November 1990, pp. 184–187

[3] H. Cho, G. Hachtel, S.W. Jeong, B. Plessier, E. Schwarz, F. Somenzi, "Algorithms for Approximate FSM Traversal," in *Proc. ACM/IEEE DAC'93*, June 1993, pp. 25–30

[4] G. Cabodi, P. Camurati, S. Quer, "Symbolic Exploration of Large Circuits with Enhanced Forward/Backward Traversals," in *Proc. IEEE EURO-DAC94*, September 1994, pp. 22–27

[5] B. Lin, A. Richard Newton, "Implicit Manipulation of Equivalence Classes Using Binary Decision Diagrams," in *Proc. IEEE ICCD'91*, October 1991, pp. 81–85

Circuit	PI	PO	FF	Gates
s208	11	2	8	96
s298	3	6	14	119
s344	9	11	15	160
s349	9	11	15	161
s382	3	6	21	158
s386	7	7	6	159
s400	3	6	21	162
s420	19	2	16	196
s444	3	6	21	181
s510	19	7	6	211
s526	3	6	21	193
s641	35	24	19	379
s713	35	23	19	393
s820	18	19	5	289
s832	18	19	5	287
s838	35	2	32	390
s953	16	23	29	418
s1196	14	14	18	529
s1238	14	14	18	510
s1423	17	5	74	657
s1488	8	19	6	653
s1494	8	19	6	647
s5378	35	49	179	2779
sbc	40	56	28	1011
m_m9	14	9	27	863

Table 1: Example statistics

Circuit	Depth	# EqC	OM t [s]	EM t [s]
s208	8	40	0.4	0.1
s298	16	8061	3.8	1.2
s344	5	18608	2.8	0.1
s349	5	18608	0.8	0.1
s382	93	608448	501.1	57.1
s386	2	15	0.2	0.0
s400	93	608448	498.7	57.0
s420	8	326	0.5	0.2
s444	93	648448	502.4	163.8
s510	6	59	1.6	0.2
s526	119	1432190	2986.4	814.8
s641	1	294912	187.4	0.3
s713	1	294912	172.9	0.3
s820	4	27	3.3	0.6
s832	4	27	3.2	0.6
s838	8	3066	3.6	3.4
s953	3	$18.4549 \cdot 10^7$	0.6	0.1
s1196	2	82944	4.1	0.9
s1238	2	82944	5.6	1.2
s1488	2	49	0.9	0.2
s1494	2	49	0.9	0.2
sbc	2	$9.40 \cdot 10^6$	-	17.8

Table 2: Comparing Lin's Equivalence Class computation to the enhanced cofactor-based method

Circuit	SCEM		SRM		
	# EqC	t [s]	# N	# EqC	t [s]
s298	15165	0.7	3	8896	0.3
s382	1001860	1.2	7	745472	2.6
s400	1001860	1.1	3	745472	9.4
s444	1001860	2.0	3	700928	2.9
s526	1992800	4.8	3	1747970	10.2
s838	440976	0.5	3	9226	0.8
s1238	82944	1.0	9	82944	0.5
$s1423^{(*)}$	-	-	12	$2.9 \cdot 10^{21}$	850.0
$s5378^{(*)}$	-	-	20	$4.0 \cdot 10^{48}$	30.4
sbc	$9.4 \cdot 10^6$	1.6	4	$9.4 \cdot 10^6$	1.0
m_m9	-	-	6	$1.3 \cdot 10^8$	19.9

Table 3: Comparing the exact method to the single cycle equivalence and the best underestimation; - means that we were unable to conclude the experiment; $^{(*)}$ means that the circuit has been simplified to an equivalent one by cofactoring it with an overestimation of the states reachable from the all-0 reset state