Sequential Equivalence Checking without State Space Traversal

C.A.J. van Eijk

Design Automation Section, Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: C.A.J.v.Eijk@ele.tue.nl

Abstract

Because general algorithms for sequential equivalence checking require a state space traversal of the product machine, they are computationally expensive. In this paper, we present a new method for sequential equivalence checking which utilizes functionally equivalent signals to prove the equivalence of both circuits, thereby avoiding the state space traversal. The effectiveness of the proposed method is confirmed by experimental results on retimed and optimized ISCAS'89 benchmarks.

1. Introduction

With the increasing use of sequential optimizations during logic synthesis, sequential equivalence checking is becoming an important practical verification problem. Conventional algorithms for solving this problem require a state space traversal of the product machine. Impressive progress has been made in this area by the introduction of so-called symbolic techniques, which are based on the application of binary decision diagrams (BDDs) to traverse the state space (see e.g. [2] for an overview). Although these techniques can conceivably handle large circuits and are still being improved (see e.g. [3]), they cannot be expected to scale well with circuit size for many types of circuits.

For combinational equivalence checking, the state-of-the-art verification methods combine a powerful base verification algorithm with techniques to exploit the structural similarities of the circuits under verification (see e.g. [7] for an overview). These similarities typically occur in practical problem instances because of the incremental nature of the design process. The techniques to capture them are based on functional equivalences, indirect implications, or permissible functions. The utilization of structural similarities has shown to be very important for the efficient verification of synthesized circuits. Because sequential optimizations such as retiming only have a restricted influence on the structure of a circuit, this approach of exploiting structural similarities is also likely to be applicable to sequential equivalence checking. In this paper, we present a new method to prove the equivalence of sequential circuits which show structural similarities. The method is not based on a state space traversal, and therefore it can handle larger circuits than existing methods.

When verifying combinational circuits, structural similarities can be proved correct before they are used to simplify the verification problem. When dealing with sequential circuits, this is clearly more difficult. In the presence of sequential feedback, it is necessary to combine the detection and utilization of similarities to really benefit from them. In this paper, we solve this problem by proposing a fixed point iteration which gradually filters sets of potentially equivalent functions until the actual equivalences remain. This filtering process only requires combinational techniques. Therefore the proposed method can be viewed as a way to extend the applicability of the state-of-the-art combinational verification techniques to sequential equivalence checking.

2. Related work

In this paper, we focus on the utilization of structural similarities to enable the verification of large sequential circuits. Alternative approaches are to put restrictions on the synthesis process, such as the C-1-D property proposed in [1], or to use formal synthesis, as described in [8].

The conventional symbolic algorithms for sequential equivalence checking do not attempt to benefit from the structural similarities of the circuits under verification. Several techniques are proposed in literature to improve them in this respect. In [6], the use of functional dependencies is proposed to exploit the relation between the state encodings of both circuits during the state space traversal of the product machine. In [12], a method is described to incrementally re-encode one of the circuits to factor out their differences.

When a sequential circuit is only optimized with combinational synthesis techniques, the correctness of the implementation can be verified with a combinational verification method if the corresponding registers in both circuits are known. An efficient technique to automatically identify this register correspondence without calculating the reachable state space of the product machine is described independently in [5] and [9]. The detection of corresponding registers also forms the basis for the utilization of structural similarities in the verification method proposed in [11]. A preprocessing step for handling retimed circuits is described in [10], which relies on 3-valued equivalence and name correspondences.

Recently, a new sequential verification method called ‘Record & Play’ was proposed in [14]. This method uses recursive learning in combination with a so-called structural fixed point iteration to find equivalent signals. By applying retiming transformations, the two circuits are made more similar. The concept of instruction queues is introduced to capture the equivalent signals.

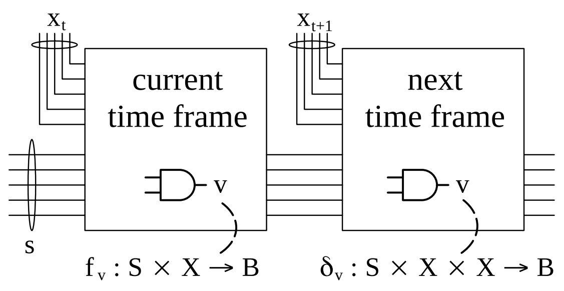
Based on the work outlined above, the following important observation can be made: In many practical cases, sequential equivalence of circuits can be proved by determining the correct relation between their state encodings rather than by calculating the entire reachable state space of the product machine. The method we propose in this paper relies on this observation. It uses a greatest fixed point iteration to identify functionally equivalent signals, which is a generalization of the techniques used in [5] and [9] for determining corresponding registers.

3. Verification method

Our basic model of a sequential circuit is a deterministic Mealy-type finite state machine (FSM) with a specified initial state. We assume that we are given two circuits, which are combined into a product machine with input space X, state space S, initial state $s_0 \in S$, next-state function $\Delta : S \times X \rightarrow S$, and output function $\Lambda : S \times X \rightarrow B$, where $B = \{0, 1\}$ denotes the set of Boolean values. The output function Λ is 1 for a given state and input vector, if all pairwise corresponding outputs of the two circuits assume the same value. The objective of sequential equivalence checking is to prove that Λ is 1 for every reachable state and input vector.

The basis of our verification method is a greatest fixed point iteration, which works on the set of all the functions of the signals in both circuits. It gradually partitions this set into equivalence classes, such that all functions in the same equivalence class are sequentially equivalent, meaning that for any reachable state and input vector, they have the same value. In each step of the iteration, two consecutive time frames of the product machine are considered which are called the current time frame and the next time frame, as shown in Figure 1. Each time frame consists of the combinational logic of the product machine. The current time frame is used to model the potential equivalences between the signals found thus far. The correctness of these equivalences is verified in the next time frame.

With each signal v, we associate a current-state function $f_v : S \times X \rightarrow B$ which expresses the value of v in the current time frame as a function of the current state and the current input vector, and a next-state function $\delta_v : S \times X \times X \rightarrow B$ which expresses the value of v in the



x_t
x_{t+1}
current time frame
next time frame
v
v
s
$f_v : S \times X \rightarrow B$
$\delta_v : S \times X \times X \rightarrow B$

Fig. 1. Time frame model of the product machine

next time frame as a function of the current state, the current input vector and the next input vector. Note that $\delta_v(s, x_t, x_{t+1}) = f_v(\Delta(s, x_t), x_{t+1})$.

Before we describe the fixed point iteration in detail, we first introduce the theory on which it is based. The set of all signals in the product machine is denoted V. Based on this set, we construct the following set of functions F. We take the initial state s_0 and a randomly selected input vector $x_0 \in X$ as a reference point. For each signal $v \in V$, we consider the value of $f_v(s_0, x_0)$. If it is 1, then we add the function f_v to the set F, and otherwise its complement $\bar{f}_v$. This procedure normalizes each signal in the product machine with respect to its polarity, which is important for the following reason: It allows the method to detect not just equivalent, but also antivalent signals (i.e., signals having opposite values in every reachable state).

Informally, the detection of structural similarities corresponds to determining a signal correspondence, i.e., identifying signals with sequentially equivalent current-state functions. This can be formalized as the calculation of an equivalence relation on F, such that all functions equivalent with respect to this relation are also sequentially equivalent. Note that this still allows a function to correspond with several other functions. Before defining the conditions an equivalence relation on F has to comply with to represent a correct signal correspondence, we first introduce the notion of a *correspondence condition* to associate a set of states with an equivalence relation on F. More specifically, this correspondence condition is a function which evaluates to true for all states conforming to that relation.

Definition 1

Given an equivalence relation $\overset{sc}{\equiv}$ on the set F. The *correspondence condition* $Q_{\overset{sc}{\equiv}} : S \times X \rightarrow B$ is the function that defines whether a state conforms to the relation $\overset{sc}{\equiv}$, i.e., whether all functions in the same equivalence class of $\overset{sc}{\equiv}$ indeed have the same value:

$$Q_{\overset{sc}{\equiv}}(s, x) = \prod_{f_m, f_n \in F \wedge f_m \overset{sc}{\equiv} f_n} f_m(s, x) = f_n(s, x) \,.$$

Note that the correspondence condition may also depend on the current input vector. This is a technicality: The theory is not influenced by adding a universal quantification over the input space in the definition of the condition.

We can now define the conditions an equivalence relation $\overset{sc}{\equiv}$ on F has to satisfy to represent a correct signal correspondence. We impose the following two conditions. The first condition is that if two functions correspond according to $\overset{sc}{\equiv}$, then they must have the same value in the initial state s_0. This guarantees that both circuits start in a state in which $Q_{\overset{sc}{\equiv}}$ holds. The second condition we impose is that if we consider two functions that correspond according to s_0, and a state in which $Q_{\overset{sc}{\equiv}}$ holds, then the associated next-state functions have to be equivalent in this state. This condition guarantees that if the two circuits are in a state for which $Q_{\overset{sc}{\equiv}}$ holds, then $Q_{\overset{sc}{\equiv}}$ will also hold for every next state of the circuits. If both these conditions are satisfied, it can directly be concluded that all functions in the same equivalence class necessarily are sequentially equivalent. We introduce the term *signal correspondence relation* to denote an equivalence relation on F that complies with the two conditions described above.

Definition 2

An equivalence relation $\overset{sc}{\equiv}$ on F is a *signal correspondence relation* iff for each pair of functions $f_m, f_n \in F$ with $f_m \overset{sc}{\equiv} f_n$:

- for all $x \in X : f_m(s_0, x) = f_n(s_0, x)$,
- for all $s \in S, x_t, x_{t+1} \in X$:
  $Q_{\overset{sc}{\equiv}}(s, x_t) \Rightarrow \delta_m(s, x_t, x_{t+1}) = \delta_n(s, x_t, x_{t+1})$.

We will use the example of Figure 2 to illustrate the notion of a signal correspondence relation and the associated correspondence condition. An example of a correct correspondence relation for this example is given by the partition $\{\{f_1\}, \{f_2\}, \{f_3, f_6\}, \{f_4, f_7\}, \{f_5\}\}$. This relation states that the signals v_3 and v_6 are sequentially equivalent, as well as the signals v_4 and v_7. The associated correspondence condition is $(v_1v_2 \equiv v_6)(v_1v_2x_t \equiv v_6x_t)$, which can be simplified to $v_1v_2 \equiv v_6$. Using the information given in the table of Figure 2, it can be checked that this correspondence relation satisfies both conditions stated in Definition 2. For example, the condition to determine whether f_3 and f_6 are equivalent, is:

$$v_1v_2 \equiv v_6 \Rightarrow \overline{v_1v_2}\,\overline{x_t} \equiv \overline{x_t + v_6} ,$$

which is a tautology. Similarly, it can be proved that the functions of the outputs v_4 and v_7 are equivalent, and thus that the two circuits are equivalent.

For a given pair of circuits, there may exist several signal correspondence relations. We state without proof that the set of all correspondence relations is a partially ordered set in which each pair of elements has a least upper bound. As a consequence, there is a unique maximum relation $\overset{msc}{\equiv}$. This maximum correspondence relation can also be characterized by the property that for any other correspondence relation $\overset{sc}{\equiv}$ and for all $f_m, f_n \in F$:

$$f_m \overset{sc}{\equiv} f_n \Rightarrow f_m \overset{msc}{\equiv} f_n . \qquad (1)$$

Our verification method uses the following theorem to prove the equivalence of sequential circuits.

Theorem 1

Let $\overset{msc}{\equiv}$ denote the maximum signal correspondence relation. If:

$$\forall s \in S, x \in X \ : Q_{\overset{msc}{\equiv}}(s, x) \Rightarrow \Lambda(s, x) ,$$

then the two circuits are sequentially equivalent.

We will now describe the greatest fixed point iteration to calculate the maximum signal correspondence relation $\overset{msc}{\equiv}$. A sequence of equivalence relations T_i is calculated, which converges to the maximum correspondence relation. The first relation T_0 compares the functions in F with respect to the initial state:

$$f_m\ T_0\ f_n \Leftrightarrow \forall x \in X : f_m(s_0, x) = f_n(s_0, x) . \qquad (2)$$

Starting with T_0, a sequence of relations is calculated by applying the second condition of Definition 2:

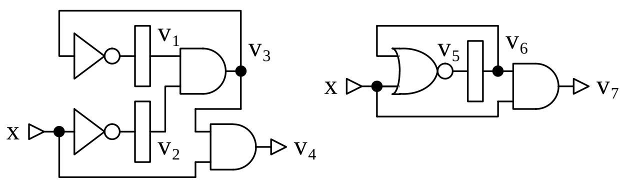
$$f_m\,T_{i+1}\,f_n \Leftrightarrow f_m\,T_i\,f_n \wedge (\forall s \in S, x_t, x_{t+1} \in X : \\ Q_{T_i}(s, x_t) \Rightarrow \delta_m(s, x_t, x_{t+1}) = \delta_n(s, x_t, x_{t+1})) . \qquad (3)$$

Since there is only a finite number of functions in F, a fixed point is reached after a finite number of iterations, i.e., at some point $T_i = T_{i+1}$. This T_i is the maximum signal correspondence relation. The number of iterations is at most $|F| + 1$, because in every iteration, except the last one, the number of equivalence classes increases by at least one.

Theorem 2

Given the sequence of equivalence relations as defined by the Equations 2 and 3. If $T_i = T_{i+1}$, then T_i is the maximum signal correspondence relation.

The proof of Theorem 2 follows a similar line of reasoning as the proof of Theorem 4.1 in [7].



init.:
$v_1 = 1$,
$v_2 = 1$,
$v_6 = 1$,
$x_t = 1$

v_i	init.	f_i	δ_i
v_1	1	v_1	$\overline{v_1v_2}$
v_2	1	v_2	$\overline{x_t}$
v_3	1	v_1v_2	$\overline{v_1v_2}\,\overline{x_t}$
v_4	1	$v_1v_2x_t$	$\overline{v_1v_2}\,\overline{x_t}x_{t+1}$
v_5	0	$x_t + v_6$	$x_{t+1} + \overline{x_t + v_6}$
v_6	1	v_6	$\overline{x_t + v_6}$
v_7	1	v_6x_t	$\overline{x_t + v_6}\,x_{t+1}$

Fig. 2. Example of a retimed and logically optimized circuit

The 'scope' of the fixed point iteration explained above is determined by the functions in the set F. Its accuracy can therefore be improved by extending this set with extra functions. In our verification method, we do this by applying forward retiming transformations to the product machine. Note that this differs from the way retiming transformations are used in [10] and [14]: we do not actually move latches (and thus we avoid the problems related to maintaining a valid initial state [13]) but rather add the extra combinational logic that would result from the retiming transformations. We only consider retimings with a lag of –1, meaning that at most one register is moved from every input to every output of a gate. This is illustrated in Figure 3. In the circuit at the left, the AND gate can be retimed with a lag of –1 by moving the registers r_2 and r_4 to the output of the gate. We model the effects of this retiming move by introducing an extra AND gate connected to the registers r_1 and r_3.

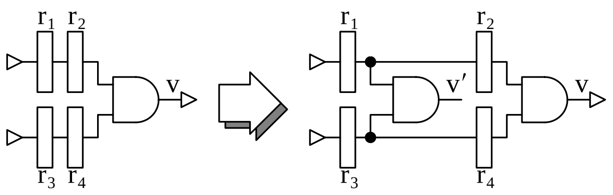
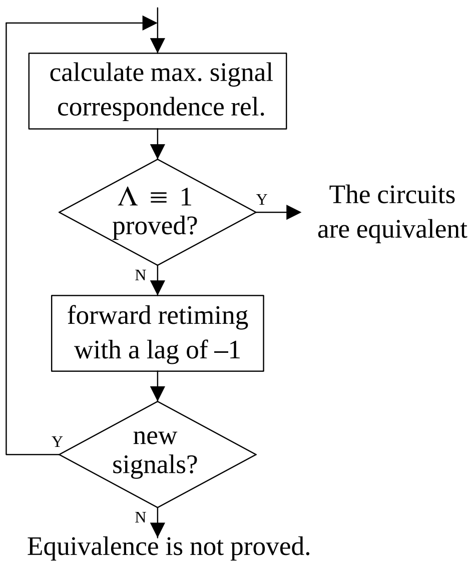


Fig. 3. Retiming with lag –1 to generate additional logic

The outline of the resulting verification method is shown in Figure 4. First the maximum signal correspondence relation is calculated. If the current-state functions of all pairwise corresponding outputs of the two circuits are equivalent according to this relation, then the sequential equivalence of both circuits is proved and we can stop. Otherwise it is checked whether the set F can be extended using retiming transformations as explained above. Note that because this step may be applied more than once, also retiming transformations with a lag smaller than –1 are considered. If the retiming generates new combinational logic and thus results in an extension of the set F, the method continues with the calculation of the maximum signal correspondence relation for this larger set of functions.


calculate max. signal correspondence rel.
Λ ≡ 1 proved?
Y
The circuits are equivalent
N
forward retiming with a lag of –1
new signals?
Y
N
Equivalence is not proved.
Fig. 4. Outline of the verification method

The proposed verification method can easily be extended to also take sequential don't cares due to the non-reachable state space into account. For example, the reachable state space of the specification can be used to strengthen the correspondence condition, i.e., instead of using the correspondence condition $Q_{\underline{\underline{sc}}}$, the condition $Q_{\underline{\underline{sc}}} \wedge S_{reach}$ can be used, where S_{reach} denotes the characteristic function of the specification's reachable state space. Note that by combining the specification's reachable state space with the correspondence condition, this information is also applied to the implementation.

Instead of using the exact reachable state space, it is also possible to use an upper bound approximation of the reachable state space, which can be calculated efficiently using techniques as e.g. described in [4].

4. Implementation issues

When implementing the method of Section 3, we have to choose an appropriate data structure for storing the relations T_i that are calculated during the refinement process. Because every T_i is an equivalence relation, it can be represented by its equivalence classes. Therefore, the choice of an appropriate data structure is not difficult: We can simply store the equivalence classes of T_i explicitly, resulting in a space complexity of $O(F)$.

In every iteration of the fixed point computation, a new relation T_{i+1} is derived from the previous relation T_i by splitting some equivalence classes into a number of smaller classes. This is done by evaluating Equation 3. The complement of the correspondence condition is basically used as a don't care set when comparing the next-state functions. We can use functional dependencies [6] of the correspondence condition to better exploit this don't care set. To illustrate this, consider the example based on the circuits of Figure 2 again. In this example, the correspondence condition is $v_1v_2 \equiv v_6$. This condition can be taken into account by actually replacing state variable v_6 by the function v_1v_2. We use a greedy heuristic based on the structure of the product machine to select the state variables which can be written as a function of other state variables before the correspondence condition is actually calculated.

Sequential simulation of the product machine with random input vectors can be used to partition the set F into sets of potentially equivalent functions; if two functions have different values during simulation, it can directly be decided that they are not equivalent. This results in a better initial approximation of the maximum signal correspondence relation, and thus reduces the required number of iterations.

5. Experimental results

This section reports the results of some preliminary experiments performed with the proposed verification method. Our current implementation constructs BDDs expressed over the input and state variables to represent the correspondence condition and the next-state functions without introducing extra variables for intermediate signals. It is based on the BDD package developed at Eindhoven University. Dynamic variable ordering is used to control the BDD variable ordering. All tests are performed on a 99 MHz HP9000/755 workstation with a memory limit of 100 (Mb) imposed on the BDD package and a time limit of 3600 (s). The verification method is tested on circuits from the ISCAS'89 benchmark set. The circuits are verified against the synthesized versions of these circuits from [14], which have been optimized by kerneling and retiming. To make these circuits more difficult to verify, we have further optimized them using script.rugged of SIS. We compare our method with the symbolic verification method of [6] which uses functional dependencies to capture the relation between the state encodings of both circuits.

Table 1 shows the experimental results. The first two columns show the name of the benchmark and the number of registers in the circuits before and after synthesis. The following columns list the run time, the maximum number of BDD nodes during verification, and the required number of iterations for both verification methods. For the proposed method, the number between parentheses in the column '#its' denotes the number of times the retiming procedure is invoked. The last column shows the percentage of signals in the specification for which a corresponding signal in the implementation is found. The average percentage of equivalences is 54%; without running script.rugged on the circuits of [14], the percentage of equivalences is 85%.

The experimental results clearly show that the proposed verification method can handle larger circuits than a symbolic method which uses BDDs to traverse the state space, even if the latter method exploits functional dependencies. If the detection of functional dependencies is disabled, the symbolic traversal method performs considerably worse. The proposed method can verify most circuits within a reasonable time, even those having a very deep state space such as s838. Only the circuits s3384 and s6669 cannot be handled, because the BDDs become too large. This problem is however more related to the combinational verification techniques used than to the proposed method.

6. Conclusion

We have proposed a new verification method for sequential equivalence checking which proves equivalence by detecting and utilizing structural similarities rather than performing a state space traversal of the product machine. A greatest fixed point iteration is used to determine sequentially equivalent signals. Because the method only requires combinational verification techniques, it is significantly more efficient than symbolic verification methods requiring a state space traversal. As the experimental results show, it can verify large circuits after extensive optimization even using plain BDD-based combinational verification techniques. We expect that the performance of the method on larger circuits can be significantly improved by applying techniques based on the introduction of extra variables representing intermediate signals (see e.g. [7] for an overview).

Although the proposed method is sound, it is not a complete method, i.e., there are pairs of equivalent circuits for which it cannot prove equivalence. The method can be used as an effective preprocessing step for a general method such as [6]. It is interesting to note that for some synthesis steps, the method is complete. This is e.g. the case for circuits optimized with combinational synthesis techniques, and also for retimed circuits. Currently we are working on further characterizing the combinations of synthesis techniques for which the method is complete.

Acknowledgements

The author would like to thank Wolfgang Kunz and Dominik Stoffel for providing the circuits of [14].

References

[1] P. Ashar, A. Gupta, and S. Malik, "Using Complete-1-Distinguishability for FSM Equivalence Checking", Proc. ICCAD, pp. 346–353, 1996.

[2] J.R. Burch, et al., "Symbolic Model Checking for Sequential Circuit Verification", IEEE Trans. on Computer-Aided Design of Integr. Circ. and Syst., vol. 13, no. 4, pp. 401–424, April 1994.

[3] G. Cabodi, et al., "Disjunctive Partitioning and Partial Iterative Squaring: An Effective Approach for Symbolic Traversal of Large Circuits", Proc. 34th DAC, pp. 728–733, 1997.

[4] H. Cho, et al., "Algorithms for Approximate FSM Traversal", Proc. 30th DAC, pp. 25–30, 1993.

[5] C.A.J. van Eijk, and J.A.G. Jess, "Detection of Equivalent State Variables in Finite State Machine Verification", Workshop notes of the 1995 IWLS, pp. 3.35–3.44, 1995.

[6] C.A.J. van Eijk, and J.A.G. Jess, "Exploiting Functional Dependencies in Finite State Machine Verification", Proc. ED&TC, pp. 9–14, 1996.

[7] C.A.J. van Eijk, "Formal Methods for the Verification of Digital Circuits", Ph.D. Thesis, Eindhoven University of Technology, September 1997.

[8] D. Eisenbiegler, R. Kumar, and C. Blumenröhr, “A Constructive Approach towards Correctness of Synthesis – Application within Retiming”, Proc. ED&TC, pp. 427–431, 1997.

[9] T. Filkorn, “Symbolische Methoden für die Verifikation endlicher Zustandssysteme,” Dissertation Institut für Informatik der Technischen Universität München, 1992.

[10] S.-Y. Huang, K.-T. Cheng, and K.-C. Chen, “On Verifying the Correctness of Retimed Circuits”, Proc. Great Lakes Symp. on VLSI, pp. 277–281, 1996.

[11] S.-Y. Huang, K.-T. Cheng, and K.-C. Chen, “AQUILA: An Equivalence Verifier for Large Sequential Circuits”, Proc. ASPDAC, pp. 455–460, 1997.

[12] S. Quer, et al., “Incremental Re-encoding for Symbolic Traversal of Product Machines”, Proc. EuroDAC, pp. 158–163, 1996.

[13] L. Stok, I. Spillinger, and G. Even, “Improving Initialization through Reversed Retiming”, Proc. ED&TC, pp. 150–154, 1995.

[14] D. Stoffel, and W. Kunz, “Record & Play: A Structural Fixed Point Iteration for Sequential Circuit Verification”, Proc. ICCAD, pp. 394–399, 1997.

Table 1. Experimental results on retimed and logically optimized circuits

circuit	#regs orig./opt.	symbolic traversal			proposed method			
		time (s)	nodes	#its	time (s)	nodes	#its	eqs (%)
s208	8 / 14	0.7	1604	256	0.4	325	8 (0)	55
s298	14 / 29	3.8	6149	19	0.8	682	5 (0)	57
s344	15 / 38	21.0	18962	7	0.6	1061	2 (0)	49
s349	15 / 38	19.2	15944	7	0.7	1272	2 (0)	49
s382	21 / 36	8.9	11339	151	3.3	1103	21 (0)	60
s386	6 / 43	7.9	8876	8	2.2	1905	5 (0)	53
s420	16 / 34	73.3	11152	65536	5.3	1725	32 (0)	55
s444	21 / 36	14.3	11035	151	3.6	1172	22 (0)	58
s510	6 / 64	2683	708940	47	2.0	2473	1 (1)	42
s526	21 / 58	62.7	44942	151	6.7	1375	27 (0)	61
s641	19 / 17	2.9	5667	3	2.1	4178	2 (0)	83
s713	19 / 17	4.7	6667	3	2.5	4196	2 (0)	83
s820	5 / 53	228	165895	11	9.6	4443	8 (0)	42
s832	5 / 57	173	115889	11	11.8	4617	8 (0)	42
s838	32 / 74	—	—	—	55.6	4548	80 (0)	55
s953	29 / 76	130	85756	10	5.7	4225	3 (0)	51
s1196	18 / 18	4.5	8236	1	3.0	4152	2 (0)	33
s1238	18 / 18	5.5	5820	1	2.9	4015	2 (0)	30
s1423	74 / 85	—	—	—	676	100830	12 (0)	53
s1512	57 / 101	—	—	—	34.7	9339	13 (0)	60
s3271	116 / 189	—	—	—	808	28499	6 (4)	26
s3330	132 / 114	—	—	—	1316	1094772	3 (1)	64
s3384	183 / 506	—	—	—	—	—	—	—
s4863	104 / 152	—	—	—	55.6	9547	2 (0)	56
s5378	179 / 263	—	—	—	801	34117	17 (1)	71
s6669	239 / 267	—	—	—	—	—	—	—