High Level Equivalence Symmetric Input Identification

Ming-Hong Su Chun-Yao Wang

Department of Computer Science
National Tsing Hua University, HsinChu, Taiwan, R.O.C

{harrysu, wcyao}@cs.nthu.edu.tw

Abstract — *Symmetric input identification is an important technique in logic synthesis. Previous approaches deal with this problem by building BDDs and developing algorithms to determine symmetric inputs. For the design whose corresponding BDDs cannot be built, BDD-based approaches cannot be applied on this problem. To avoid the limitations of BDD-based approaches, simulation-based methods have been proposed. It is applicable to designs described in arbitrary level, especially to high-level and black box designs. Previous simulation-based approaches focus on determining the inputs of nonequivalence symmetry. In this paper, we propose a simulation-based approach to identify equivalence symmetric inputs. The experimental results on a set of ISCAS-85 and MCNC benchmarks are also presented.*

I. INTRODUCTION

Symmetry input identification is to find the symmetric relation of all inputs. Symmetric input sets are the subsets of inputs. Grouping symmetric inputs to form symmetric input sets and thus any permutations of the inputs within a subset leave the function invariant. The problem to find the maximal symmetric inputs sets has been formulated in [3] [4]: Given a function $f(x)$, find maximal subsets of inputs $x_1, x_2, ..., x_n \subseteq X$, such that $x_1 \cup x_2 \cup ... \cup x_n = X$ and the inputs in every x_i can be permuted in any fashion without changing the functionality.

Method to finding the maximal symmetric input sets is based on finding all pairs of symmetric inputs and then take the unions of all the pairs having nonempty intersection. Previous approaches [2[3][6] are based on checking the equality of cofactors of the function. The maximal symmetric input sets could be computed after checking all symmetric pairs. However, those approaches are very time-consuming and not feasible for large functions. While the number of inputs is large, representing functions using BDDs will improve the efficiency of cofactor computation. A simple symmetry test is to check whether the BDD representations of two cofactor functions are isomorphic or not. This can be seen in Figure 1. However, computing multiple cofactor pairs is still expensive for large functions. This is because the repeated computation leads to create and delete a large number of intermediate BDD nodes. Previous approaches based on BDDs and Boolean functions are summarized as follows. [6] avoids redundant cofactor computation by some criterions and thus speed up the computation process. An efficient algorithm without computing cofactors is proposed in [2]. [10] uses a K-Disjointness Paradigm which can compute disjoint situations with Hamming distance K between Boolean function to find the maximum symmetries. [11] formulates symmetry identification as an equation without using cofactor computation and equivalence checking.

This work was supported in part by the National Science Council of R.O.C. under Grant NSC94-2220-E-007-041.

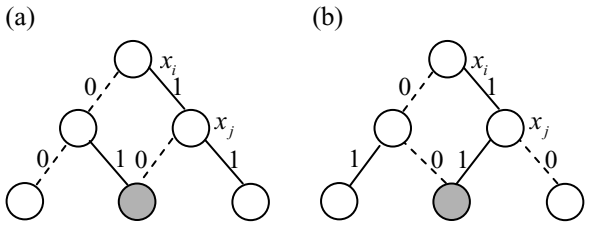

Figure 1: a) Nonequivalence Symmetry
b) Equivalence Symmetry

In addition to BDD-based and Boolean function-based methods, simulation-based approaches were applied to circuits without having compact BDDs or Boolean functions representation. [5] establishes two stages to accomplish the symmetric input identification and using a simulation-based method as the first stage of its two-stage algorithm.

Most of previous works focus on determining *nonequivalence* (NE) symmetry. For NE symmetry, symmetric pair is an equivalence relation. However, this is not true for *equivalence* (E) symmetry. For example, three inputs $\{ x_i , x_j , x_k \}$ could be distinguished as NE symmetric inputs by any two symmetric pairs are held because of transitivity. But for E symmetry, three pairs of inputs have to be symmetric simultaneously. Consequently, determining E symmetry needs more efforts than determining NE symmetry. This paper proposes a simulation-based approach to determining E symmetry.

The remainder of the paper is organized as follows. In the next section we briefly overview different types of symmetries and introduce the representation of maximal symmetric input sets. Our algorithm will be presented in Section 3. The experimental results of E symmetry are

0-7803-9451-8/06/$20.00 ©2006 IEEE.

presented in Section 4. Finally, Section 5 concludes the paper.

II. PRELIMINARIES

This section reviews different types of symmetries at first. Then, we introduce a representation for maximal symmetric input sets. Finally, we show the naïve approach to identifying symmetric inputs.

A. *Overview of Symmetries*

A *cofactor* of a function $f(x)$ with respect to variables x_i and x_j is the function resulting from the substitution into $f(x)$ of specific values for x_i and x_j. For example, the cofactor of $f(x)$ with respect to $x_i = 0$ and $x_j = 1$ is $f(x_1, ..., 0, ..., 1, ..., x_n)$, which is denoted as $f_{\bar{x}_i x_j}$ [8].

For any pairs of variables x_i and x_j, there are four cofactors, which are $f_{\bar{x}_i \bar{x}_j}$, $f_{\bar{x}_i x_j}$, $f_{x_i \bar{x}_j}$, and $f_{x_i x_j}$. Different categories of symmetries can be defined according to the equality of two cofactors among them. A function $f(x)$ exhibits a nonequivalence (NE) symmetry in inputs x_i and x_j, if $f_{\bar{x}_i x_j} = f_{x_i \bar{x}_j}$. When $f_{\bar{x}_i \bar{x}_j} = f_{x_i x_j}$, the function is said to exhibit equivalence (E) symmetry with respect to x_i and x_j. The illustrations of NE and E symmetries are shown in Figure 2 and Figure 3, respectively.

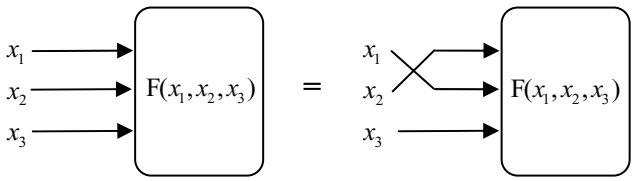

Figure 2: Illustration of nonequivalence symmetry

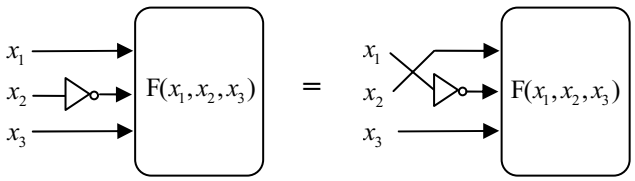

Figure 3: Illustration of equivalence symmetry

B. *Symmetric-ASymmetric Inputs (SASIs) Representation*

For an N-input circuit, there are C_2^N symmetric pairs of all inputs. The maximal symmetric input sets could be computed after checking all symmetric pairs. A naïve way to presenting the result is to construct an $N \times N$ triangular matrix for an N-input circuit. Each entry in the triangular matrix shows the symmetry of the corresponding inputs x_i and x_j except the diagonal entries. This representation is very simple but hard to understand globally. Therefore, we use *Symmetric -ASymmetric Inputs* (SASIs) [9][1], which is an *implicit* representation to present the maximal symmetric input sets. By the SASIs, if any two inputs are not in the same group, then they are asymmetric inputs. Otherwise they are "possibly" symmetric. For an N-input circuit, we number the inputs from 1 to N. Initially, we can assume that all inputs are possibly symmetric, so the SASIs representation is (1 2 3 …. N). If we can confirm that input *i* is asymmetric to the other inputs, the SASIs representation is (*i*) (1 2 …. *i*-1 *i*+1 N). The following example demonstrates the details of the SASIs representation.

Example 2.1: Given a 10-input circuit, the inputs are numbered from 1 to 10. Initially, we assume all inputs are possibly symmetric and thus the corresponding SASIs representation is (1 2 3 4 5 6 7 8 9 10). While the SASIs representation is (1 2 3 4 5) (6 7 8) (9) (10), it indicates that input 9 and input 10 are asymmetric to the other inputs. If SASIs representation could be divided into ten groups, then we claim that all inputs are asymmetric inputs.

C. *Naïve Approach*

A pair of patterns whose assignments are identical except on inputs x_i and x_j, and $x_i = x_j$ (00 or 11) in each pattern, is capable of distinguishing whether x_i and x_j are E symmetric or not. These pairs of patterns are called *legal pattern pairs*.

There are 2^{N-2} legal pattern pairs for any two inputs in an N-input circuit. Two inputs are E symmetric while the outputs of each legal pattern pair are identical. Otherwise they are E asymmetric inputs. We illustrate it using the following example.

Example 2.2: For a 5-input circuit, if we want to recognize whether input 1 and input 2 are E symmetric inputs or not. We have to exhaustively simulate $2^{5-2} = 8$ legal pattern pairs, which are {(00000, 11000), (00100, 11100), (00010, 11010), (00001, 110001), (00110, 11110), (00101, 11101), (00011, 11011), (00111, 11111)}. If the outputs of each legal pattern pairs are identical, input 1 and input 2 are symmetric. Otherwise they are asymmetric.

It is clear that to identify two inputs are E asymmetric is easier than to identify they are E symmetric. Thus, our approach will target at the identification of asymmetric inputs.

Definition 2.1: A pair of inputs x_i and x_j is denoted as *VP* (x_i , x_j) if they have not been recognized as symmetric or asymmetric.

The naïve approach is an exhaustive approach. It simulates all legal pattern pairs to recognize whether the targeted VP is asymmetric or not. If there exits one legal pattern pair with different outputs, then the process to recognizing the targeted VP would cease.

III. EQUIVALENCE SYMMETRY IDENTIFICATION ALGORITHM

Since the naïve approach needs a great number of

patterns and comparisons to identify symmetric inputs. A heuristic which applies two sets of patterns to recognize all VPs simultaneously is investigated.

Definition 3.1 [1]: For an N-input combination circuit, the set consists of all patterns with *m* 1s and (*N*-*m*) 0s is denoted as θ_m^N, where $m \in [0, 1, 2, \ldots, N\text{-}1, N]$. The size of θ_m^N is the number of patterns in θ_m^N and is denoted as $|\theta_m^N|$ and $|\theta_m^N|$ equals C_m^N, where

$$C_m^N = \frac{N!}{(N-m)! \times m!}$$

Following equations represent the relations of θ_m^N for different *m* and *N*:

$$|\theta_m^N| \le |\theta_{m+1}^N| \text{ for } m \in \{1, 2, \ldots, \lfloor (N\text{-}1)/2 \rfloor\}$$
$$|\theta_m^N| \le |\theta_{m-1}^N| \text{ for } m \in \{\lceil (N+1)/2 \rceil, \ldots, N\text{-}1\}$$

Theorem 3.1: For any two pattern sets $\{\theta_i^N, \theta_{i+2}^N\}$, those two pattern sets can be used to recognize all VPs.

However, this heuristic is infeasible while *i* increases. For example, considering two pattern sets $\{\theta_1^{100}, \theta_3^{100}\}$, there are C_1^{100}=100 patterns in θ_1^{100}, and C_3^{100}=161,700 patterns in θ_3^{100}. It conducts 100×161,700 = 16,170,000 comparisons for identifying E symmetry.

The heuristic approach fails due to a great number of patterns and comparisons have to be generated and conducted. The number of patterns in a pattern set depends on two factors, one is the length of a pattern, i.e., the number of inputs. The other is the number of 1s in a pattern of the pattern set. Since the pattern set is determined by the number of 1s, it seems difficult in reducing this factor. Therefore, we attempt to divide inputs into as many groups as possible. Therefore, an improved approach is introduced.

Definition 3.2[1]: A *multiple element group* (MEG) is a group that contains more than one element in the SASIs representation. A *single element group* (SEG) is a group that contains only one element.

The improved approach aims at each MEG and generates the corresponding pattern sets for each MEG. While the size of MEG is reduced, the number of pattern in a pattern set could also be reduced.

Example 3.1: For a 10-input circuit and assume the SASIs representation is (1 2 3 4 5) (6 7 8 9 10) after generating $\{\theta_1^{10}, \theta_3^{10}\}$. The second step is to generate $\{\theta_2^{10}, \theta_4^{10}\}$ and we have to generate $(C_2^{10} + C_4^{10})$ = 255 patterns and conduct $(C_2^{10} \times C_4^{10})$ = 9,450 comparisons by using the heuristic approach. But in considering the improved approach, for the MEG (1 2 3 4 5), it only generates $\{\theta_2^5, \theta_4^5\}$. It is the same to the MEG (6 7 8 9 10). The total number of patterns and comparisons by using the improved approach are $(C_2^5 + C_4^5) \times 2$ = 30 and $(C_2^5 \times C_4^5) \times 2$ = 100, respectively. As compared with the heuristic approach, the improved approach is effective in diminishing the total number of patterns and comparisons.

If the size of MEG is large, the number of patterns to be generated is still large. Thus, next we will propose an algorithm that systematically generates smaller number of patterns to distinguish as many E-asymmetric inputs as possible.

Definition 3.3: The *distance* of VP(x_i, x_j) in an MEG is the difference of relative position of x_i and x_j.

Theorem 3.2: For an MEG with *K* elements, the number of VPs with distance *i* is (*K* - *i*) and the maximal distance among all VPs is (*K* - 1).

Example 3.2: For an MEG (2 3 5 6 7 8 9), we number the position from left to right as 1 to 7. Please note the distance of a VP is the difference of relative position. While the initial position (position number is 1) or the allocation of elements in an MEG is changed, the distance of VPs would also be changed. All VPs in the MEG are listed by their distances in Table I.

Table I The distance of VPs

Distance	VP	|VP|
1	(2,3),(3,5),(5,6),6,7),(7,8),(8,9)	6
2	(2,5),(3,6),(5,7),(6,8), (7,9)	5
3	(2,6),(3,7),(5,8),(6,9)	4
4	(2,7),(3,8),(5,9)	3
5	(2,8),(3,9)	2
6	(2,9)	1

Definition 3.4: For an N-input circuit, *circular pattern set* for an MEG with *K* elements is the set that consists of all patterns which satisfy following conditions in θ_m^N and is denoted as $\alpha_{m,i}^K$.

1). Initial position is circularly set in each element of the MEG.
2). The distance of elements assigned value 1 is 1 except on the last two elements. The distance of the last two elements assigned value 1 is *i*.
3). If *m*=1, then *i* is 1.

Example 3.3: For a 10-input circuit, there is an MEG with 7 elements and assume the SASIs representation is (2 3 5 6 7 8 9). The circular pattern set $\alpha_{3,1}^7$ is {0110100000, 0010110000, 0000111000, 0000011100, 0000001110, 0100000110, 0110000010}. To represent the patterns concisely, a simple representation that indicates which inputs are assigned 1 is used. Hence the simple representation of $\alpha_{3,1}^7$ is {(2,3,5), (3,5,6), (5,6,7), (6,7,8), (7,8,9), (8,9,2), (9,2,3)}. The circular pattern set $\alpha_{3,2}^7$ is {0110010000, 0010101000, 0000110100, 0000011010, 0100001100, 0010000110, 0100100010} and the corresponding simple representation is {(2,3,6), (3,5,7), (5,6,8), (6,7,9), (7,8,2), (8,9,3), (9,2,5)}.

Theorem 3.3: For an MEG with *K* elements, a couple of circular pattern sets $\{\alpha_{m,1}^K, \alpha_{m+2,i}^K\}$ can be used to recognize VPs with distance *i* and *(K - i)*.

Proof: The distances of patterns in $\alpha_{m,1}^N$ is 1, hence the distance sequence is $(x_1 = 1, x_2 = 1, \ldots, x_{i-1} = 1)$ for all patterns in $\alpha_{m,1}^N$. The patterns in $\alpha_{m+2,i}^N$ with the distance sequence $(x_1 = 1, x_2 = 1, \ldots, x_{i-1} = 1, x_i = 1, x_{i+1} = i)$ and the patterns in $\alpha_{m,1}^N$ could be used to recognize VPs with distance *i*. Since *i* represent the distance of last two critical 1s and we regards the distance as circular, VPs with

distance *(N - i)* could be treated as *i*. Therefore, $\{ \alpha^N_{m,1}, \alpha^N_{m+2,i} \}$ could be used to recognize VPs with distance *i* and (*N - i*). ■

Now, we will explain how to utilize circular pattern set to recognize symmetric inputs. Considering an MEG with *K* elements, there are C_2^K VPs and could be divided into (*K-1*) sets by the distance. Since a couple of $\{ \alpha^K_{m,1}\ \alpha^K_{m+2,i} \}$ could be used to recognize VPs with distance *i* and (*K - i*), and the possible distance of all VPs is from 1 to (*K - 1*), we apply the following rules to recognize all VPs.

Rule 1: Choosing circular pattern set $\alpha^K_{m,1}$ in θ^N_m.

Rule 2: Choosing circular pattern sets $\alpha^K_{m+2,i}$ in θ^N_{m+2} where, i =1, 2,…, $\lceil (N-1)/2 \rceil$ in θ^N_{m+2}.

Example 3.4: For a 10-input circuit and we assume the SASIs representation is (2 3 5 6 7 8 9)(1)(4)(10) after generating and comparing $\{ \theta^{10}_0, \theta^{10}_2 \}$. Next we choose circular pattern set $\alpha^7_{1,1}$ in θ^{10}_1 and $\alpha^7_{3,1}$ in θ^{10}_3 for the MEG (2 3 5 6 7 8 9). Those two circular pattern sets can be used to recognize VPs with distance 1 and 6. This can be seen in Figure 4. Similarly, circular pattern set $\alpha^7_{3,2}$ and $\alpha^7_{3,3}$ could be used to recognize VPs with distance 2 and 5 as well as 3 and 4 as comparing $\alpha^7_{1,1}$, respectively. Those two illustrations could be seen in Figure 5 and Figure 6. It is obvious that those three circular pattern sets can cover all distances of all VPs.

$\alpha^7_{1,1}$	$\alpha^7_{3,1}$	VPs	Distance
2	2 3 5	(3, 5)	1
3	3 5 6	(5, 6)	1
5	5 6 7	(6, 7)	1
6	6 7 8	(7, 8)	1
7	7 8 9	(8, 9)	1
8	8 9 2	(9, 2)	6
9	9 2 3	(2, 3)	1

Figure 4: Comparing ($\alpha^7_{1,1}$, $\alpha^7_{3,1}$) covers VPs with distance 1 and 6

$\alpha^7_{1,1}$	$\alpha^7_{3,2}$	VPs	Distance
2	2 3 6	(3, 6)	2
3	3 5 7	(5, 7)	2
5	5 6 8	(6, 8)	2
6	6 7 9	(7, 9)	2
7	7 8 2	(8, 2)	5
8	8 9 3	(9, 3)	5
9	9 2 5	(2, 5)	2

Figure 5: Comparing ($\alpha^7_{1,1}$, $\alpha^7_{3,2}$) covers VPs with distance 2 and 5

$\alpha^{/}_{1,1}$	$\alpha^{/}_{3,3}$	VPs	Distance
2	2 3 7	(3, 7)	3
3	3 5 8	(5, 8)	3
5	5 6 9	(6, 9)	3
6	6 7 2	(7, 2)	4
7	7 8 3	(8, 3)	4
8	8 9 5	(9, 5)	4
9	9 2 6	(2, 6)	3

Figure 6: Comparing ($\alpha^7_{1,1}$, $\alpha^7_{3,3}$) covers VPs with distance 3 and 4

Figure 7 shows the flow chart that we proposed for finding maximal symmetric inputs sets. Our approach reads a design with arbitrary levels and generates patterns. The results of patterns provide information to the remaining VPs. Grouping all remaining VPs to from the updated SASIs. Then further heuristic patterns are generated and simulated again by the updated SASIs in the next iteration. If all inputs are recognized as asymmetric or the iterations are over the bound, our approach will be terminated and the maximal symmetric input sets will be returned.

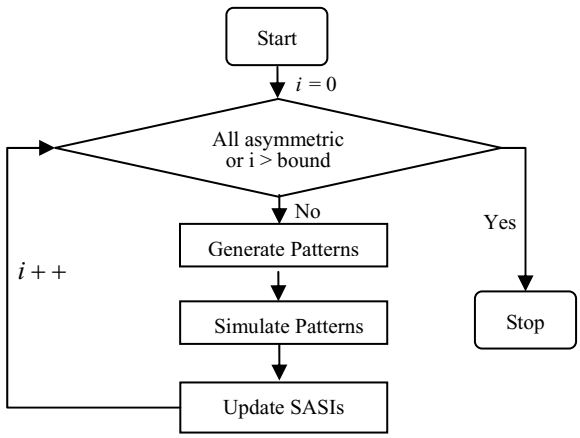

Figure 7: The flow chart of our approach

IV. EXPERIMENTAL RESULTS

We have implemented the proposed algorithm in Verilog HDL. Experiments are conducted over a set of ISCAS-85 and MCNC benchmarks which are described in Verilog HDL.

We compare the experimental results with [10]. [10] is an BDD-based approach for E symmetry identification. It claims that all VPs can be identified exactly.

Table II summarizes the experimental results of [10] and ours. The first column shows the name of each benchmark and the following two columns **#in** and **#out** represent the number of inputs and outputs. The following columns show the CPU time measured in second and the results. In [10], the time for building BDDs was not listed. We construct the BDDs for each benchmark by CUDD package [7] without using any reordering technique and its time is shown in the "**reading**" column on a "SUN SPARC II" workstation measured in second. The last column shows the number of variable pairs that cannot be recognized as asymmetric inputs by [10] and our approach. According to Table II, our CPU time is less than that of [10] with including the time of BDD construction, and our results are the same with [10] for most benchmarks. In c2670 and c7552, however, our approach returns more VPs that cannot be recognized as asymmetric than [10], but the CPU time is less than that of [10]. Note that our approach is applicable to the designs

whose compact BDDs cannot be built. For example, c6288 is a multiplier design, one cannot have an efficient BDD representation for it. Hence, the proposed approach is a robust approach to some degree.

V. CONCLUSIONS

Random simulation could also find some asymmetric VPs. But it may generate redundant patterns for some recognized asymmetric VPs. Thus, simulation with randomly generated patterns is inefficient. In this paper, we propose a systematic patterns search algorithm for computing maximal symmetric inputs sets. It is applicable to designs described in arbitrary level, especially to high-level and black box designs. Experimental results on ISCAS-85 and MCNC benchmarks demonstrate the effectiveness and efficiency of our approach.

REFERENCES

[1] C.-L Chou, C.-Y Wang, G.-W Lee, and J.-Y Jou, "Graph automorphism-based algorithm for determining symmetric inputs," *in Proceedings of IEEE International Conference on Computer Design*, pp. 417-419, 2004.

[2] A. Mishchenko, "Fast computation of symmetries in Boolean functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, pp. 1588-1593, Nov. 2003.

[3] E. J. McCluskey, "Detection of group invariance or total symmetry of a Boolean function," *Bell System Technology Journal,* pp. 1445-1453, Nov. 1956.

[4] E. J. McCluskey, *Logic Design Principles with Emphasis on Testable Semicustom Circuits, Prentice-Hall*, 1986.

[5] I. Pomeranz and S. M. Reddy, "On determining symmetries in inputs of logic circuits," in *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 500-507, 1993.

[6] C. Scholl, D. Moller, and P. Molitor, "BDD minimization using symmetries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 81-100, Feb. 1999.

[7] F. Somenzi, "CUDD: CU Decision Diagram Package, Release 2.3.1," University of Colorado at Boulder, 2001.

[8] C.-C. Tsai and M. Marek-Sadowska, "Generalized Reed-Muller forms as a tool to detect symmetries," *IEEE Transactions of Computers*, vol. 45, pp. 33-40, Jan. 1996.

[9] C.-Y Wang, S.-W Tung, and J.-Y Jou, "On Automatic Verification Pattern Generation for SoC with Port Order Fault Model", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, pp. 466-479, Apr. 2002.

[10] K.-H. Wang and J.-H. Chen, "Symmetry Detection for Incompletely Specified Functions with K-Disjointness Paradigm," *in Proceedings of the IEEE Asia and South Pacific Design Automation Conference,* vol. 2, pp. 994-997, 2005.

[11] K.-H. Wang and J.-H. Chen, "Symmetry detection for incompletely specified functions," *in Proceedings of IEEE Design Automation Conference*, pp. 434-437, 2004.

Table II
Experimental results of the equivalence symmetric input identification

circuit	#in	#out	time (s)			symmetry pair	
			reading	[10]	ours	[10]	ours
c880	60	26	11.57	0.03	1.75	0	0
c1355	41	32	1.30	0.05	0.68	0	0
c1908	33	25	--	--	0.28	--	0
c432	36	7	--	--	0.19	--	0
c499	41	32	1.17	0.05	0.66	0	0
c3540	50	22	18.96	0.08	2.42	0	0
c5315	178	123	>1hr	0.02	49.38	0	0
c2670	233	140	>1hr	0.08	593.04	28	227
c7552	207	108	>1hr	0.17	633.61	6	160
c6288	32	32	--	--	0.25	--	0
des	256	245	3.42	0.03	14.36	0	0
rot	135	107	3.08	0.07	26.31	1	1
9sym	9	1	--	--	0.46	--	0
alu4	14	8	--	--	0.71	--	0
cordic	23	2	--	--	2.89	--	0
t481	16	1	--	--	0.98	--	0