A Testability Measure for Hierarchical Design Environments

Mike H.C. Lee and D.L. Tao

Department of Electrical Engineering
SUNY at Stony Brook, Stony Brook, NY 11794

Abstract

In this paper a new approach is proposed to compute testability of a combinational circuit in a hierarchical design environment. The testability of a circuit is first computed at the functional level using the Walsh expression of the functional block, and its complexity is linear with respect to the number of functional blocks. The functional level testability measure is then used to compute the testability at the gate/switch level. Our extensive simulation results show that the testability measure of the proposed method reflects closely to the actual testability measure (both at the functional level and the gate level) when the granularity of a functional block is much higher than that of primitive gates.

1 Introduction

In a hierarchical design environment, VLSI circuits are first designed and simulated at the functional level, and then a gate/switch level implementation will be generated by a silicon compiler or automatic synthesizing tools. Meanwhile, testing of a VLSI circuit must be done at the gate level in order to ensure the confidence of testing. It is known that design for test at the functional level can significantly reduce the complexity of testing at the gate level. Hence, it is necessary to develop testing tools at the functional level which can be used to alleviate the testing problems at the gate level. Testability measures have been playing a critical role in all automatic testing environments. A number of gate level testability measures have been proposed, but they cannot be used at the functional level [1, 3, 4, 8, 12]. On the other hand, a few functional testability functions are developed, but they do not address the testing problem at the gate level [2, 11]. The objective of our research is to bridge this gap by developing a testability measure for hierarchical design environments, i.e., it can be used both at the functional level and the gate/switch level.

In this paper a new hierarchical testability measure is introduced for combinational circuits in hierarchical design environments. The testability measure is constructed based on the Walsh expression of a logic function and COP [1]. A Walsh expression is suitable for functional level testing because the Walsh expression itself is a functional representation and Walsh coefficients reflect inherent features of the logic function. The strategy used in COP is utilized to make computational time and memory requirement practically feasible. Moreover, the testability measure at the functional level will be used to compute testability at the gate/switch level. Furthermore, our extensive simulation results demonstrate the proposed method is able to provide accurate testability measures for combinational circuits, and so it can be used to guide design for testability at both the functional and the gate levels.

2 Walsh Series

The most common way to represent a logic function is to use a Boolean equation. A logic function can also be expressed by other representations, such as a Rademacher-Walsh expression (or simply Walsh expression) as follows.

$$F(\mathbf{x}) = \sum_{\forall i \subseteq I} C_i \varphi_i(\mathbf{x}), \quad (1)$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, $\varphi_i(\mathbf{x}) = \prod_{\forall l \in i}(2x_l - 1)$, and $I = \{1, 2, \cdots, n\}$. This set of polynomials is orthonormal with respect to each other under the inner product:

$$\frac{1}{2^n} \sum_{\mathbf{x} \in \mathbf{X}} \varphi_i(\mathbf{x}) \varphi_j(\mathbf{x}) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases},$$

where $\mathbf{X}$ is the set containing all possible $\mathbf{x}$. As a result, Walsh coefficients can be computed as follows:

$$C_i = \frac{1}{2^n} \sum_{\mathbf{x} \in \mathbf{X}} F(\mathbf{x}) \varphi_i(\mathbf{x}). \quad (2)$$

We now use an example to illustrate a function being represented by a Walsh expression.

Example 1: Let us consider the circuit shown in Figure 1. The Walsh coefficients for $f(x_1, x_2, x_3, x_4, x_5)$ are shown in Table 1. Hence, the output of the circuit can be represented by:

$$F(x_1, x_2, x_3, x_4, x_5) = \sum_{\forall i \subseteq \{1,2,3,4,5\}} C_i \varphi_i(\mathbf{x}). \quad (3)$$

For example, $\varphi_0(\mathbf{x}) = 1$, $\varphi_2(\mathbf{x}) = (2x_2 - 1)$, $\varphi_{1,3,5}(\mathbf{x}) = (2x_1 - 1)(2x_3 - 1)(2x_5 - 1)$. When $x_1x_2x_3x_4x_5 = 11110$, by using Equation 3 and Table 1, we find $F(1, 1, 1, 1, 0) = 0$, whereas when $x_1x_2x_3x_4x_5 = 01010$, we find $F(0, 1, 0, 1, 0) = 1$. □

A Walsh expression has been used in testing [9, 10], where Walsh coefficients are used as signatures in

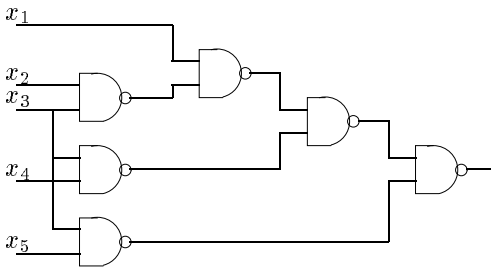

Figure 1: A Simple Combinational Circuit

built-in self test circuit design. In this paper, we will show that a Walsh expression is a perfect representation for computing functional-level testability. The signal probability of a logic function and the observability of an input line can be computed by using the following theorems.

Theorem 1: When all input variables of a logic function are statistical independent of each other, the signal probability of $F(\mathbf{x})$, $S(F(\mathbf{x}))$, can be computed as follows

$$S(F(\mathbf{x})) = \sum_{\forall i \subseteq I} C_i \prod_{\forall l \in i} (2S(x_l) - 1), \quad (4)$$

$S(x_l)$: the signal probability at x_l.

Theorem 2: When all input variables of a logic function are statistical independent of each other, the observability of an input line can be computed using the following expression.

$$S(O_k) = \sum_{\forall i \subseteq I-k} \hat{C}_i \prod_{\forall l \in i} (2S(x_l) - 1). \quad (5)$$

where $\hat{C}_i = \sum_{\forall l \subseteq I-j} 4 \times C_l \times C_{i \oplus l}$, and $i \oplus l$ represents the *symmetric difference* of two sets i and l.

The proof can be found in [7]. A drawback of using the Walsh expression is that the number of terms grows exponentially with the number of input variables. When a circuit is represented at the functional level, it is reasonable to assume that each functional block has less than or equal to 10 primary inputs, such as adders, encoders/decoders, multiplexers, etc, and so the testability function constructed based on a Walsh expression will not introduce a significant computational overhead. This is also justified by the fact that majority of cell libraries at the RT level have 10 or fewer input variables. For a logic function with more than 10 input variables, the testability of this function can be computed by partitioning the input space, such as multipliers, data path, multiplexers.

3 Determining Testability Hierarchically

Given a circuit which is represented by an interconnection of functional blocks, the signal probabilities at input/output nets of functional blocks can be computed recursively from the primary inputs. In such a process, we assume that signals on inputs of a functional block are statistically independent of each other [1]. In addition, for a fanout net its observability OB_{fn} is computed as

$$OB_{fn} = 1 - \prod_{for\ all\ fanout\ branches} (1 - OB_{fb}), \quad (6)$$

where OB_{fb} is the observability on a fanout branch. We now present a procedure to compute the detection probabilities of a circuit represented at the functional level. Similar to the definitions of levelizing a circuit at the gate level, we define input/output distance for each functional block as follows. When a circuit is represented at the functional level, all primary input lines have the input distance ($ID(x_i)$) as 0. The input distance of a functional block i is $n+1$ if $ID(j) < n+1$ for all immediate predecessors j of i and $ID(k) = n$ for some immediate predecessor k of i. Similarly, the output distance of a primary output is 0. The output distance of a functional block i is $n+1$ if $OD(j) < n+1$ for all immediate successor j of i and $OD(k) = n$ for some immediate successor k of i.

Procedure I: Functional-Level Detection Probability

1. Assign signal probabilities to all primary inputs and $i = 1$.
2. For each and every functional block with $ID(i)$, compute signal probabilities at its output(s) using Equation 4. Note that the signal probability of an input line is equal to the one at the output line of the predecessor.
3. If there exists a primary output whose signal probability is unknown, then $i = i + 1$ and go to Step 2.
4. Assign the observabilities to primary output lines and $i = 1$.
5. For each and every functional block with $OD(i)$, compute observability for each output line j, OB_j, using Equation 6. For each input line of a functional block, k, compute the relative observability with respect to each output line of the functional block, $ROB_{k,j}$ (Equation 5). The observability of an input line, k, is equal to

$$OB_k = 1 - \prod_{\substack{\forall\ output \\ lines,\ j,\ of\ a\ block}} (1 - ROB_{k,j} \times OB_j).$$

Moreover, the detection probability of a net is equal to the product of its signal probability and its observability.

[1] The validity of this assumption will be discussed in Section 4.

6. If there exists a primary input line whose observability is unknown, then $i = i + 1$ and go to Step 5. □

Example 2: Let us consider the circuit in Figure 2, and the signal probabilities at the primary inputs are $\frac{1}{2}$, and the observability at the output is 1. The signal probabilities and observabilities of the input/output lines of functional blocks are computed by using Walsh expressions in Equations 4 and 5. We obtain $S(l_3) = \frac{3}{8}$, $S(l_4) = \frac{33}{64}$, $OB(l_3) = \frac{3}{8}$, and $OB(l_4) = 1$. The observability at the input x_2 with respect to net $l_3 = \frac{3}{4}$, and so the observability at the input x_2 with respect to the primary output $l_4 = \frac{9}{32}$. Similarly, the observability at the input x_4 with respect to the primary output $l_4 = \frac{11}{32}$. □

The signal probabilities and the observabilities at the input/output nets of functional blocks provide us a testability measure of a circuit at the functional level. This can be carried out at the gate level or at the switch level. In other words, the testability can be computed **hierarchically** as follows. The detection probability of a fault inside of a block can be computed by using the signal probability at the input lines of the block and the observability at the output line(s) of the block. This is done by the following procedure.

Procedure II: Gate/Switch Level Detection Probability

1. Set the detection probabilities of all faults into 0.
2. Partition the functional blocks of the circuit into m groups, and two functional blocks are in the same group iff they have the same gate-level (switch-level) implementation. Set i to 1.
3. In group i let l be the number of input variables of the functional block and n be the number of elements. Set j to 0.
4. Let v_j be an input vector where j is the decimal representation of v_j. Find all faults which are detectable by v_j in a functional block of group i at each of its outputs. Let $k = 1$.
5. For the k-th functional block at the i-th group, compute the probability of the input vector, $Pr_k(v_j)$, using signal probabilities of input lines of the k-th block, the detection probability of a fault f in the k-th block will be computed by

$$DP_f = DP_f + Pr_k(v_j) \times \left(1 - \prod_{\substack{\forall q \mid v_j \text{ detects} f \\ \text{at the output line } q}} (1 - OB_q)\right).$$

6. $k = k + 1$. If $k \leq n$, then go to Step 5.
7. $j = j + 1$. If $j < 2^l$, then go to Step 4.
8. $i = i + 1$. If $i \leq m$, then go to Step 3. □

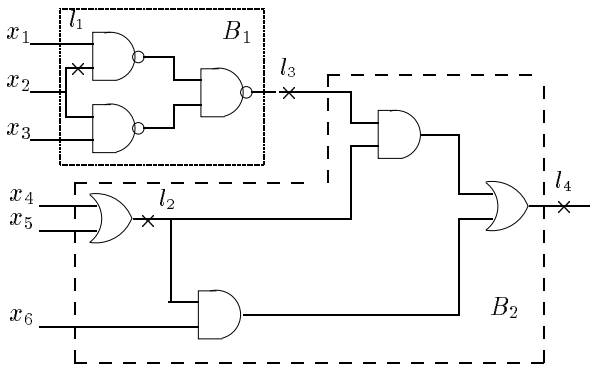

Figure 2: A Circuit Represented in Hierarchical Fashion

Example 3: Let us continue from the previous example. We have $S(l_3) = \frac{3}{8}$, $S(l_4) = \frac{33}{64}$, $OB(l_3) = \frac{3}{8}$, and $OB(l_4) = 1$. The detection probability of f_1 (stuck-at 0 at l_1) with respect to the inputs and output of block B_1 is $\frac{1}{8}$, and the observability of l_3 is $\frac{3}{8}$. Hence, the detection probability of this fault will be $\frac{3}{64}$. Similarly, the signal probability of l_3 is $\frac{3}{8}$. The test patterns for detecting f_2 in block B_2 include: $l_3 x_4 x_5 x_6 = 0001, 1000$, and 1001. The probabilities of receiving these patterns at the inputs of B_2 are respectively equal to $\frac{5}{8} \times \frac{1}{8}$, $\frac{3}{8} \times \frac{1}{8}$ and $\frac{3}{8} \times \frac{1}{8}$. Using Equation 4 and 5, we found that the detection probability of f_2 (stuck-at 1 at l_2) is $\frac{11}{64}$. □

4 Experimental Results

We have implemented Procedure I and Procedure II in C language and run it on Sun-Unix systems. We first consider a set of FPGA benchmark circuits by Kuznar *et al* [6]. In [6], 10 ISCAS benchmark circuits are respectively mapped into Xilinx-2000 FPGAs. An Xilinx-2000 series FPGA contains $64 \sim 100$ Configurable Logic Blocks (CLB), and each CLB can implement a 4-input logic function. In [6] 10 ISCAS benchmark circuits represented as an interconnection of CLBs will be utilized as functional descriptions of ISCAS85 benchmark circuits. For the sake of simplicity, we call them *ISCAS functional level benchmark circuits*. Table 2 shows the results of mapping ISCAS benchmark circuits into Xilinx FPGAs. It includes the number of gates and the number of fan-out stems (FOS) in the underlying gate level implementation. This table also includes the number of CLBs and the number of fan-out stems at the functional block level.

We consider all single stuck-at faults at the input/output lines of each functional block, and compute their detection probabilities. The Procedure I is used to compute the detection probability of each fault in 10 *ISCAS functional-level benchmark circuits*, denoted by $DP_c(f)$. In order to verify the accuracy of our results, a large number of random test vectors ($32k$

$\sim$ 1 M) are applied to those *ISCAS functional-level benchmark circuits*. By using a functional-level fault simulator recently developed at the SUNY at Stony Brook, the actual detection probability of each fault is computed and it is denoted by $DP_a(f)$. For each fault f, we compare $DP_a(f)$ with $DP_c(f)$, and then compute the root mean square (RMS) between the computed detection probability and actual detection probability as follows.

$$RMS = \sqrt{\sum_{\forall f} \Big(DP_c(f) - DP_a(f) \Big)^2 / \ M},$$

where M is the total number of faults in a circuit. We now consider the detection probability at the gate level for all single stuck-at faults. The detection probability of each stuck-at fault is computed using Procedure I and Procedure II. Again, a large number of test vectors are applied, and a gate level fault simulator is used to compute an actual detection probability. Moreover, the RMS between the computed detection probability and actual detection probability is shown in Table 3. In Table 3, we also list corresponding results computed using COP.

Similar to COP, we have assumed that signals at input lines of a functional block are statistically independent of each other. At the gate level, this assumption introduces large computational errors because of reconvergent fan-out circuits. From Table 3, we can see that if the number of fanout stems at the functional level is very close to that at the gate level (except c1355), then the performance of the proposed scheme is comparable to COP. We design four sets of additional experiments. Seven functional circuits are constructed and considered. The RMS values of detection probability at the functional level and at the gate level are shown in Table 4. From these experiments it can be seen that when the granularity of a functional block is much higher than primitive gates, the proposed method generates a much more accurate testability measure than COP.

Another important question is how close our functional testability measure it is to the actual testability. This is equivalent to the problem of testing the closeness of two sets of data. The correlation coefficient (CC) is widely used for this purpose. The correlation coefficient is defined as follows:

$$CC = \frac{n \sum_{\forall i \in F} AD_i \cdot CD_i - \sum_{\forall i \in F} AD_i \sum_{\forall i \in F} CD_i}{\sigma_{AD} \sigma_{CD}},$$

where AD_i is the actual detection probability for the fault i which is obtained from fault simulation, CD_i is the detection probability obtained by the proposed method for fault i, n is total number of faults, F is the set containing all faults, $\sigma_{AD} = \sqrt{n \sum_{\forall i \in F} AD_i^2 - (\sum_{\forall i \in F} AD_i)^2}$ and $\sigma_{CD} = \sqrt{n \sum_{\forall i \in F} CD_i^2 - (\sum_{\forall i \in F} CD_i)^2}$. Now we consider the faults at the functional level (all single stuck-at faults at the input/output of functional blocks) and compute the testability using Procedure I. Also we consider the same set of faults of a gate level implementation and apply 1 M samples to fault simulator and compute the detectability for these faults. The results are given in Table 5. It can be seen that the functional level testability information obtained from the proposed method reflects the actual testability obtained from the gate level. When the granularity of a functional block is much larger than primitive gates, the average correlation coefficient is equal to 0.99, i.e., the prediction error range is about 1%. Therefore our method is suitable for a hierarchical design environment when the detectability information is needed and the gate level implementation is not even available.

COP assumes that signals at the inputs of a gate are independent of each other, and so a considerable amount of errors is introduced in computing the testability. Recently, Ercolani *et al* [3] proposed two new methods to compute the testability at the gate level, and a limited signal dependency is taken into account. The average RMS values using these two methods are 78% and 87% of that by COP as shown in Table 6. If the granularity level of a functional block is much higher than primitive gates, then the average RMS values using the proposed method is only 31% of that from COP (using results in Table 5).

Since each functional block has up to 10 inputs, the run time for computing testability of a functional block is about several CPU seconds in our experiments using a SUN Sparc-IPC workstation. The execution times of our implementations are shown in Table 7, they are measured when our programs are running on a SUN Sparc-IPC workstation. Since our implementation is not optimized, the execution time in Table 7 can be further reduced by incorporating heuristics, such as decomposition technique used in [5]. More importantly, the run time complexity of the proposed scheme is linear in terms of the number of blocks, and so our technique is applicable to large combinational circuits.

5 Conclusion

In this paper, a new testability measure has been proposed for a hierarchical design environment. To our best knowledge, this is the first comprehensive study on hierarchical testability measurement. Since the testability is computed in top-down fashion, it can be easily intergrated into the existing hierarchical CAD design environments. We found that when the granularity of functional blocks is much larger than that of primitive gates, more accurate testability can be obtained than existing approximation methods, such as COP and that [3], provided that the computational complexity is linear with respect to the number of blocks in a given circuit. The functional level testability computed by using the proposed method is very close to the actual testability measure at the functional level. Moreover, the gate-level testability

measure of the proposed technique is more accurate than that obtained using approximated methods, such as COP and that in [3].

References

[1] F. Brglez, P. Pwnall, and R. Hum, "Application of testability analysis: from ATPG to critical delay path tracing", *Proceedings of IEEE International Test Conference*, pp. 705-712, 1984.

[2] C.H. Chen and P.R. Menon, "An approach to functional level testability analysis", *Proceedings of International Test Conference*, pp. 373 - 379, 1989.

[3] S. Ercolani, *et al* "Testability measures in pseudorandom testing", *IEEE Transactions on CAD*, Vol. CAD-11, no. 6, pp. 794-800, June, 1992.

[4] S.K. Jain and V.D. Agrawal, "Statistical fault analysis" *IEEE Design and Test of Computers*, Vol. 2, no. 3, pp. 38-49, Feb. 1985.

[5] R. Krieger, B. Becker, and R. Sinkovic, "A BDD - based algorithm for computation of exact fault detection probabilities", *Proceedings of 23rd FTCS*, pp. 186-195, June, 1993.

[6] R. Kuznar, F. Brglez, and K. Kozminski, "Cost Minimization of Partitions into Multiple Devices", *IEEE/ACM Proceedings of the 30th DAC*, Dallas, Texas, June 14-18, 1993.

[7] Mike H.C. Lee, "Applying Walsh Series to Automatic Testing" Ph.D Dissertation, Dept. of Electrical Engineering, SUNY at Stony Brook, 9/1994.

[8] S.C. Seth *et. al.*, "An exact analysis for efficient computation of random-pattern testability in combinational circuits", *Proceedings of 16-th FTCS*, pp. 318-323, 1986.

[9] M. Serra and J.C. Muzio, "Space compaction for multiple-output circuits", *IEEE Transactions on CAD*, Vol. CAD-7, no. 10, pp. 1105-1113, October, 1988.

[10] A.K. Susskind, "Testing by verifying Walsh coefficients", *IEEE Transactions on Computers*, Vol. C-32, no. 2, pp. 198-201, Feb., 1983.

[11] K. Thearling and J. Abraham, "An easily computed functional level testability measure", *Proceedings of International Test Conference*, pp. 381 - 390, 1989.

[12] H. J. Wunderlich, "Protest: a tool for probabilistic testability analysis", *Proceedings of 22nd DAC conference*, pp. 204-211, 1985.

Table 1. Walsh Coefficients of a Logic Function

c_0	c_5	$c_{2,3}$	$c_{4,5}$	$c_{1,3,5}$	$c_{3,4,5}$	$c_{2,3,4,5}$
$\frac{19}{32}$	$\frac{5}{32}$	$\frac{1}{32}$	$\frac{3}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{32}$
c_1	$c_{1,2}$	$c_{2,4}$	$c_{1,2,3}$	$c_{1,4,5}$	$c_{1,2,3,4}$	$c_{1,2,3,4,5}$
$\frac{-9}{32}$	$\frac{1}{32}$	$\frac{-1}{32}$	$\frac{1}{32}$	$\frac{-1}{32}$	$\frac{-1}{32}$	$\frac{1}{32}$
c_2	$c_{1,3}$	$c_{2,5}$	$c_{1,2,4}$	$c_{2,3,4}$	$c_{1,2,3,5}$	
$\frac{1}{32}$	$\frac{7}{32}$	$\frac{-1}{32}$	$\frac{-1}{32}$	$\frac{-1}{32}$	$\frac{-1}{32}$	
c_3	$c_{1,4}$	$c_{3,4}$	$c_{1,2,5}$	$c_{2,3,5}$	$c_{1,2,4,5}$	
$\frac{3}{32}$	$\frac{1}{32}$	$\frac{-3}{32}$	$\frac{-1}{32}$	$\frac{-1}{32}$	$\frac{1}{32}$	
c_4	$c_{1,5}$	$c_{3,5}$	$c_{1,3,4}$	$c_{2,4,5}$	$c_{1,3,4,5}$	
$\frac{-3}{32}$	$\frac{1}{32}$	$\frac{5}{32}$	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{-1}{32}$	

Table 2. 1985-ISCAS Functional-Level Benchmark Circuits

	Circuit	c432	c499	c880	c1355	c1908
Gate Level	NG	133	170	239	488	293
	NFOS	39	26	50	226	74
Xilinx	NCLB	63	74	110	74	147
	NFOS	39	26	50	42	73
	Circuit	c2670	c3540	c5315	c6288	c7552
Gate Level	NG	530	821	1182	2384	1538
	NFOS	133	253	192	1393	409
Xilinx	NCLBs	210	373	535	833	611
	NFOS	123	246	190	1393	401

where NG, NFOS, and NCLB respectively denote the number of gates, the number of FOS, the number of CLBs.

Table 3. RMS of Detection Probabilities

Circuit	c432	c499	c880	c1355	c1908
Cop	0.084	0.033	0.029	0.126	0.045
FL	0.099	0.045	0.028	0.019	0.052
GL	0.085	0.041	0.028	0.024	0.024
Circuit	c2670	c3540	c5315	c6288	c7552
Cop	0.077	0.060	0.039	0.214	0.064
FL	0.059	0.049	0.032	0.258	0.053
GL	0.053	0.046	0.028	0.232	0.048

Table 4. RMS of Detection Probabilities

Circuit	FL NB	GL NG	GL RMS COP	FL RMS New	GL RMS New
16x16 Mul.	49	2068	0.412	0.056	0.040
Mux16	5	35	0.037	0.012	0.006
Mux64	21	147	0.023	0.010	0.004
LZC16	8	65	0.026	0.030	0.015
LZC64	19	190	0.009	0.014	0.006
MAX16	15	87	0.076	0.035	0.030
MAX64	75	435	0.097	0.030	0.024

Table 5. Detection Probability Correlations

Circuit	CC	Circuit	CC
c432	0.860	16-bit Multiplier	0.968
c499	0.984	16x1 Multiplexer	0.997
c880	0.984	64x1 Multiplexer	0.994
c1355	0.997	16-bit LZC	0.997
c1908	0.972	64-bit LZC	0.998
c2670	0.960	MAX16	0.995
c3540	0.952	MAX64	0.977
c5315	0.983		
c6288	0.723		
c7552	0.960		

Table 6. RMS Improvements Over COP

$\frac{RMS_{DWAA}}{RMS_{COP}}$ [3]	$\frac{RMS_{CCM}}{RMS_{COP}}$ [3]	$\frac{RMS_{NEW}}{RMS_{COP}}$
0.78	0.87	0.31

Table 7. Run Time for Computing Detection Probabilities

Circuit	c432	c499	c880	c1355	c1908
GL (sec)	6.63	6.86	11.1	34.6	14.4
FL (sec)	0.53	0.35	0.80	0.42	0.92
Circuit	c2670	c3540	c5315	c6288	c7552
GL (sec)	32.0	42.7	73.8	101.5	84.4
FL (sec)	1.42	2.62	3.73	5.69	4.18

Circuit	16x16 Mul.	Mux16	Mux64	LZC16
GL (sec)	380.8	1.45	1.72	1.90
FL (sec)	91.8	0.10	0.2	0.13
Circuit	LZC64	MAX16	MAX64	
GL (sec)	19.7	30.3	33.9	
FL (sec)	5.8	2.80	4.62	

Note that GL (FL) denotes gate-level (functional level).