

A Partitioning Method for Achieving Maximal Test Concurrency in Pseudo-Exhaustive Testing*

Rajagopalan Srinivasan, Charles A. Njinda and Melvin A. Breuer

Department of Electrical Engineering – Systems
University of Southern California
Los Angeles, CA 90089-0781, U.S.A.

Abstract — Pseudo-exhaustive testing of combinational circuits usually requires multiple test sessions and/or more than a minimum number of test signals, i.e. unique input sequences. In this paper we present a methodology for partitioning combinational circuits such that they can be pseudo-exhaustively tested with a minimal number of test signals in a single test session. Circuits are logically partitioned during test mode and unrelated inputs are combined to achieve maximal test concurrency.

Keywords: Pseudo-exhaustive testing, test session, test signal, maximal test concurrency

1 Introduction

Exhaustive testing of a combinational circuit with large number of inputs is very time consuming. To reduce test time without compromising on the coverage of the stuck-at faults, the circuit can be tested pseudo-exhaustively. In pseudo-exhaustive testing, the circuit is partitioned into subcircuits, referred to as *segments*, and each segment is tested exhaustively. The number of inputs to any segment is restricted to some predetermined value. A minimum number of special segmentation cells [1, 2] can be placed in the original circuit to form the segments. These cells are transparent in normal mode and act as pseudo-inputs and pseudo-outputs during test mode. The segments need not be disjoint and each segment can have multiple outputs.

For an (n', m') circuit, n' -input, m' -output circuit – exhaustive testing with $2^{n'}$ patterns may be prohibitive for large value of n' . The circuit can be partitioned such that no segment depends on more than k inputs, where k is usually a user-defined parameter and $k < n'$. Each segment can be exhaustively tested with at most 2^k patterns. In the test mode, the original (n', m') circuit gets modified to an (n, m, k) circuit – n -input, m -output circuit, and each output depends

on at most k -inputs, where $k < n' \leq n$ and $m' \leq m$. This is illustrated in example 1.

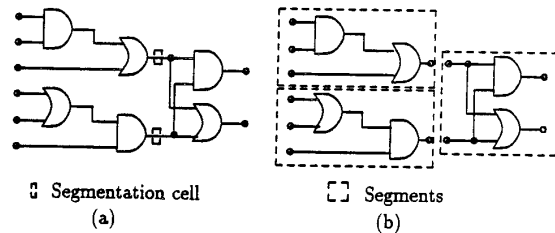


Figure 1: A partitioned circuit in (a) normal mode (b) test mode

Example 1 Consider the $(6, 2)$ circuit shown in Figure 1(a). The circuit is partitioned such that each segment depends on at most three inputs. Two segmentation cells are used to partition the circuit. In the test mode the circuit is modified to be a $(8, 4, 3)$ circuit. The modified circuit along with the segments are shown in Figure 1(b). □

One method of pseudo-exhaustive testing, namely *verification testing* [3], partitions the circuit into output cones. An *output cone* is a segment that includes all the gates and inputs feeding the output. Thus each segment has exactly one output. For an (n, m, k) circuit all output cones depend on at most k inputs. The circuit is partitioned into m segments. A *test signal* is the unique sequence of binary values applied at a circuit input. An output cone can be exhaustively tested with k test signals. In this paper we will consider only segments formed by output cones.

The segments of an (n, m, k) circuit can be exhaustively tested in two ways. One way is to use multiple test sessions. The output cones of the circuit can be grouped into k -testable groups [4]. All the cones in a k -testable group can be exhaustively tested simultaneously with k test signals. The upper bound on the number of test sessions is $\lceil m/2 \rceil$, since any two arbitrary cones can be tested simultaneously.

Another way is to test all segments in a single test session. The number of test signals required is less

*This work was supported by the Defense Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under Contract No. JFBI90092. The views and conclusions considered in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U. S. Government.

than or equal to n . Two inputs are said to be **related** if there exists at least one output that depends on both of them; else they are said to be **unrelated**. Unrelated circuit inputs can be applied the same test signals in the test mode. In general, the total number of test signals required can be reduced from n to w where $k \leq w \leq n$. Circuits requiring only k test signals are referred to as **maximal test concurrency (MTC)** circuits [3]. In this paper, we will restrict our attention to pseudo-exhaustive testing using a single test session.

Generation of an optimal test set for pseudo-exhaustive testing of an (n, m, k) non-MTC circuit is a hard problem. An optimal test set is a minimal test set that contains an exhaustive set of patterns for each output cone. The greatest lower bound on this number of patterns is 2^k . Test patterns for non-MTC circuits can be generated by various means: (a) counter based methods including the use of a constant weight counter [5], or k -bit counter and n 2-bit multiplexer [6] (b) LFSRs based on primitive polynomials [1], linear codes [7] and cyclic codes [8].

To reduce the number of patterns, verification testing [3] employs a minimum number of test signals. For a (n, m, k) circuit, the number of test signals required can be reduced from n to w where $k \leq w \leq n$. Universal verification test sets can be determined for different values of w . Test sets are not guaranteed to be optimal for $w > k + 1$.

It should be noted that an optimal test set can be easily generated for an (n, m, k) MTC circuit. A k -bit counter can exhaustively test all output cones. Therefore we attempt to modify non-MTC circuits during test mode to achieve maximal test concurrency. Since the circuit outputs depend only on a maximum of k -inputs, it may be possible to test the circuit exhaustively with just k test signals by modifying the circuit in the test mode.

The paper is organized as follows. The motivation for this work, circuit models and definitions with notation are presented in Section 2. Section 3 describes the partitioning algorithm. Experimental results and conclusions are presented in Sections 4 and 5 respectively.

2 Preliminaries

2.1 Motivation

In order to test all segments in a single test session, an (n, m, k) non-MTC circuit requires w test signals, where $k < w \leq n$. However, for testing the segments exhaustively, all 2^w patterns are not necessary. Only those subsets of k -out-of- n combinations of inputs on which the output cones depend need be tested exhaustively. This leads to designing complicated test pattern generators based on LFSR and coding theories. On the other hand, if we restrict the number of test signals to k , multiple test sessions are needed. This is illustrated in example 2.

Example 2 Consider a $(3, 3, 2)$ non-MTC circuit shown in Figure 2(a). To exhaustively test all three

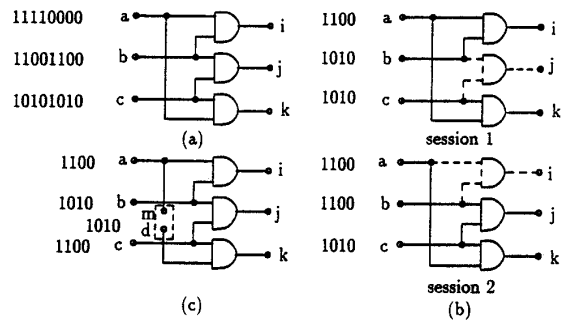


Figure 2: Pseudo-exhaustive testing of a non-MTC circuit (a) in a single test session (b) in multiple test sessions (c) modified to an MTC circuit.

output cones (segments) in a single test session, three test signals are required. The test signals consists of all 2^3 patterns. However an optimal test set containing just 2^2 patterns is $\{000, 011, 101, 110\}$. If the number of test signals is restricted to $k (= 2)$, two test sessions are required as shown in Figure 2(b). In the first session, output cone j is ignored and inputs b and c share the same test signal. The output cones i and k are tested exhaustively. In the next session inputs a and b share the same test signal, output cone i is ignored and output cones j and k are tested exhaustively. The circuit can be modified to achieve maximal test concurrency by adding a segmentation cell. The modified $(4, 4, 2)$ MTC circuit in the test mode is illustrated in Figure 2(c). Input d and output m are the newly formed pseudo-input and pseudo-output respectively. Inputs a and c , and b and d are unrelated and share the same test signals. \square

The partitioning procedures mentioned earlier modify an (n', m') circuit to an (n, m, k) circuit. If the modified (n, m, k) circuit is an MTC circuit, it can be tested with k test signals in a single test session. The patterns can be generated very easily without any complicated test pattern generators. However the partitioning procedures do not guarantee that the modified circuit is an MTC circuit. Therefore the modified circuit has to be further partitioned to achieve maximal test concurrency. This demands more modification to the original circuit in terms of additional segmentation cells. In this work, we investigate the feasibility of modifying non-MTC circuits to be MTC circuits. There exists a trade off between area overhead due to segmentation cells and test time due to multiple test sessions.

2.2 Circuit Model

The combinational circuit is modeled as a directed acyclic graph. The nodes represent inputs, outputs and gates. The interconnection signals are represented by edges. The segments are defined by the output cones of the circuit. Every output cone of the circuit forms a subgraph. The subgraphs need not be disjoint.

A **bridge** of a graph is an edge whose removal will leave the graph disconnected [9]. In other words, a bridge forms a cutset by itself. For the circuit, a **breakpoint** is a logical cut of a signal that removes the dependencies of all inputs feeding the signal from the output(s) fed by the signal. A breakpoint is implemented as a segmentation cell. An edge common to one or more subgraphs is said to be a **candidate** for a breakpoint if it forms a bridge for all subgraphs associated with it. A breakpoint has the capability to make inputs unrelated, i.e. they do not exist within any single output cone. Unrelated inputs can share the same test signal in test mode. The concepts are illustrated in example 3.

Example 3 Consider the (3, 3, 2) non-MTC circuit shown in Figure 2(a). The circuit can be modified to an MTC circuit with one breakpoint. This breakpoint is implemented as a segmentation cell as shown in Figure 2(c). It removes the dependency of input a from the output k . In other words, inputs a and c are made unrelated. Inputs a and c , and b and d can thus share the same test signal. \square

Primary inputs without fanout, and primary outputs of gates without fanout are trivial candidates. These trivial candidates are not considered for breakpoints as they do not contribute to any solution. Among other candidates, a minimum number of candidates are selected and implemented as breakpoints to achieve maximum test concurrency. \square

2.3 Procedure Outline

The partitioning procedure for converting an (n, m, k) non-MTC circuit to an MTC circuit is outlined below:

1. Determine the output cones of the circuit.
2. Identify all candidates for inserting breakpoints by analyzing the directed graph model of the circuit.
3. Partition the circuit to achieve maximum test concurrency by inserting a minimum number of breakpoints. The resulting unrelated inputs are combined with each other and share the same test signals.

The partitioned circuit is an MTC circuit and can be exhaustively tested in one test session with 2^k patterns. This method provides a systematic way for modifying a non-MTC circuit to be an MTC circuit during test mode. Unfortunately not all non-MTC circuits are modifiable due to the non-availability of suitable breakpoints. For circuits without reconvergent fanout, there is at most only one path from any input to any output. Thus all signals in the circuit are candidates for breakpoints.

It should be noted that only bridges are considered as candidates for breakpoints because of their unique characteristics. Otherwise multiple signals have to be

considered together as candidates to remove the dependency of selected inputs from appropriate outputs. This will lead to exponential computational complexity in terms of number of signals.

2.4 Definitions and Notation

For an (n, m, k) circuit, let I_j denote the j th input, $1 \leq j \leq n$, and O_i denote the i th output $1 \leq i \leq m$.

Definition 1 Dependency matrix (D-matrix)
 $D_{m \times n}$ is a binary matrix with elements

$$d_{ij} = \begin{cases} 1 & \text{if } O_i \text{ depends on } I_j, \\ 0 & \text{otherwise.} \end{cases}$$

The weight of input I_j is $\sum_{i=1}^m d_{ij}$. \square

The summation $\sum_{j=1}^n d_{ij}$ denotes the number of inputs on which the output O_i depends. Since any output depends on at most k inputs, $\sum_{j=1}^n d_{ij} \leq k$, for $i = 1, \dots, m$. The weight of an input denotes the number of outputs dependent on it.

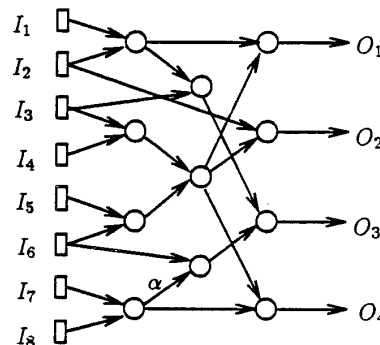


Figure 3: Graph model of an (8, 4, 6) circuit.

Example 4 Let us consider the graph model of an (8, 4, 6) circuit shown in Fig 3. The D-matrix and the input weights are shown in Table 1. \square

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
O_1	1	1	1	1	1	1	0	0
O_2	0	1	1	1	1	1	0	0
O_3	1	1	1	0	0	1	1	1
O_4	0	0	1	1	1	1	1	1
weight	2	3	4	3	3	4	2	2

Table 1: Dependency matrix

Definition 2 Relational matrix (R-matrix)

$R_{n \times n}$ is a symmetric matrix consisting of binary vector elements $r_{ij} = b_1 b_2 \dots b_m$ where

$$b_x = \begin{cases} 1 & \text{if } O_x \text{ depends on both } I_i \text{ and } I_j, \\ 0 & \text{otherwise.} \end{cases}$$

The weight of relational vector r_{ij} is $\sum_{x=1}^m b_x$. \square

The weight of relational vector r_{ij} denotes the number of outputs that depend on both the inputs I_i and I_j . The R-matrix for example 4 is given in Table 2. Inputs I_i and I_j are related if and only if the weight of relational vector $r_{ij} > 0$.

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
I_1	-	1010	1010	1000	1000	1010	0010	0010
I_2		-	1110	1100	1100	1110	0010	0010
I_3			-	1101	1101	1111	0011	0011
I_4				-	1101	1101	0001	0001
I_5					-	1101	0001	0001
I_6						-	0011	0011
I_7							-	0011
I_8								-

Table 2: Relational matrix

Definition 3 A candidate represented by $\alpha = (I_p \dots I_q)_{O_1 \dots O_j}$ corresponds to a common bridge for the cones of the outputs O_1, \dots, O_j dependent on inputs I_p, \dots, I_q . The size of candidate α is $|\{I_p, \dots, I_q\}|$. The weight of candidate α is $\sum_{I_x \in \{I_p, \dots, I_q\}} (\text{weight of } I_x)$, where the summation is over all $I_x \in \{I_p, \dots, I_q\}$. \square

Example 5 Consider the graph of Fig 3. The edge α shown forms a bridge for the subgraph formed by output cone O_3 and depends on inputs I_7 and I_8 . Since this edge is not common to any other subgraph, it is a candidate for a breakpoint. The candidate α is represented as $(I_7 I_8)_{O_3}$. The size of the candidate α is 2 and its weight is 4 ($= 2 + 2$). \square

3 Partitioning for Achieving MTC

3.1 Merging

Two inputs are said to be merged if they share the same test signal in test mode. Unrelated inputs can be merged in test mode. Merging an input pair reduces the number of test signals by one. For an (n, m, k) circuit, the number of test signals must be reduced from n to k to achieve MTC. Hence $(n - k)$ input pairs have to be merged. The results on the merging concept are summarized below.

Lemma 1 Inputs I_i and I_j related only by output O_k can be merged if and only if there exists a candidate for output O_k that includes exactly one of the inputs.

Proof: (If) Assume there exists a candidate for output O_k that includes exactly one of the inputs I_i or I_j . Selection of this candidate as a breakpoint ensures output O_k depends on exactly one of these inputs. These inputs become unrelated and therefore can be merged. *Note:* If the candidate depends on neither I_i nor I_j , then selecting it as a breakpoint still makes O_k depend

on both the inputs. Hence inputs I_i and I_j cannot be merged. On the other hand, if the candidate depends on both the inputs, then selecting it as a breakpoint ensures O_k depends on neither I_i nor I_j . However, the newly formed pseudo-output will depend on both the inputs. Hence inputs I_i and I_j cannot be merged.

(Only if) Suppose there exists no candidate for output O_k that includes exactly one of the inputs. No other candidate can make the inputs unrelated. Therefore the inputs cannot be merged. \square

For the general case, the necessary and sufficient condition to merge an input pair is given by the following theorem.

Theorem 1 Inputs I_i and I_j related by $r_{ij} = b_1 b_2 \dots b_m$ can be merged if and only if for every $b_x \neq 0$, there exists a candidate for output O_x that includes exactly one of the inputs.

Proof: Case 1: Weight of $r_{ij} = 0$

Inputs I_i and I_j are unrelated and therefore can be merged.

Case 2: Weight of $r_{ij} > 0$

(If) Suppose for every $b_x \neq 0$, there exists a candidate for output O_x which includes exactly one of the inputs. Selection of these candidates as breakpoints ensures that none of the outputs depend on both the inputs. Hence the inputs become unrelated and can be merged.

(Only if) Suppose there exists a $b_x \neq 0$ such that no candidate for output O_x includes exactly one of the inputs. No other candidate can ensure that O_x does not depend on both the inputs. Hence the inputs cannot be unrelated. Therefore inputs I_i and I_j cannot be merged. \square

Corollary 1 Merging of a related input pair requires at least one breakpoint.

Proof: An input pair can be merged only if they are unrelated. If the input pair is related by more than one output, more than one breakpoint may be required to make them unrelated. Thus a breakpoint need not necessarily make any input pair unrelated. \square

3.2 Procedure

We shall now describe a formal procedure for modifying an (n, m, k) circuit, where $k < n$, to achieve maximal test concurrency.

procedure mtc():

1. Form D-matrix $D_{m \times n} = [d_{ij}]$ and determine the weights of the inputs.
Form R-matrix $R_{n \times n} = [r_{ij}]$ and determine the number of zero relational vectors.
2. While there exists a zero relational vector r_{ij} ,
merge inputs I_i and I_j by modifying the D and R matrices.
Let p inputs be merged. To reduce the number of test signals to k , $(n - k - p) = q$ (say) additional inputs must yet be merged.

3. Model the circuit as an directed acyclic graph and determine candidates for breakpoints. Let N be the total number of candidates. Sort the candidates according to their sizes in the following order: $q, (q+1), (q+2), \dots, (k-1), k, (q-1), (q-2), \dots, 2, 1$. For candidates with the same size, sort in increasing order of their weights.
4. Let $n \leftarrow 1$ be the number of breakpoints considered at a time.
5. Select a new set of n candidates from the ordered candidate list.
For each candidate selected
 - (a) assume it is implemented as a breakpoint. A breakpoint produces a new pseudo-input and pseudo-output.
 - (b) include the pseudo-input and pseudo-output in the D and R matrices.
6. Merge all possible input pairs by modifying the D and R matrices after every merge. All of the newly created pseudo-inputs must be merged with original inputs. Let x of the original set of inputs be merged.
7. If $q = x$ then exit with success; else
 - (a) restore original D and R matrices as of step 3.
 - (b) if all possible sets of n candidates had been considered, increment n by one.
 - (c) if $n > N$ then exit with failure; else go to step 5.

Explanation: For an (n, m, k) circuit, where $k < n$, the number of test signals must be reduced from n to k . Since merging an input pair reduces the number of test signals by one, the total number of input pairs to be merged is $(n - k)$. Unrelated inputs can be merged with each other.

For merging with input I_i , whenever there are choices of other inputs, the one with the maximum weight is selected. This heuristics helps in merging maximum possible input pairs. Merging of inputs I_i and I_j collapses their columns into one in the D -matrix. The binary entries in the original columns are *or*-ed to create entries for the collapsed column. The R -matrix is also modified accordingly.

A candidate of size q has the capability of making at most $2q$ input pairs unrelated. Thus at most q inputs can be merged. However inputs associated with a candidate cannot be merged with each other. Among two candidates with equal size, the one with lower weight is preferred since inputs with lower weights can be merged more easily. Inputs of weights m can be ignored as all outputs depend on them. They cannot be merged with other inputs.

When a candidate is selected as a breakpoint, it creates a new pseudo-input and pseudo-output. The

number of rows and columns in the D and R matrices are increased by one. The length of all relational vectors is also incremented by one.

A solution with a minimal number of breakpoints is sought. Hence all possible sets of n candidates are considered with n ranging from 1 to N .

Complexity: The computational complexity of determining candidates is linear in terms of number of signals in the circuit. The selection of minimal set of breakpoints is exponential in terms of number of candidates. The special way of sorting the candidates helps in determining a minimal solution much faster. However a greedy heuristics can be employed to obtain non-minimal solutions. Such a heuristic is not presented in this paper.

Example 6 We shall illustrate the partitioning procedure with the $(8, 4, 6)$ circuit whose graph model is shown in Figure 3.

1. The D and R matrices are given in Tables 1 and 2 respectively. The number of zero relational vectors is zero.
2. The number of inputs merged (p) is 0.
The number of inputs yet to be merged ($n - k - p = q$) is 2.
3. Since the circuit has no reconvergent fanouts, all the signals are candidates for breakpoints. The total number of candidates (N) is 17. The candidates sorted according to their sizes and weights are:
Size = 2: $(I_7 I_8)_{O_3}, (I_7 I_8)_{O_4}, (I_1 I_2)_{O_1}, (I_1 I_2)_{O_3},$
 $(I_3 I_4)_{O_1 O_2 O_4}, (I_5 I_6)_{O_1 O_2 O_4}.$
Size = 3: $(I_6 I_7 I_8)_{O_3}, (I_1 I_2 I_3)_{O_3}.$
Size = 4: $(I_3 I_4 I_5 I_6)_{O_1}, (I_3 I_4 I_5 I_6)_{O_2},$
 $(I_3 I_4 I_5 I_6)_{O_4}.$
Size = 1: $(I_2)_{O_1 O_3}, (I_2)_{O_2}, (I_3)_{O_1 O_2 O_4}, (I_3)_{O_3},$
 $(I_6)_{O_1 O_2 O_4}, (I_6)_{O_3}.$
4. The number of breakpoints considered at a time (n) is 1.
5. The selected candidate for breakpoint is $\alpha = (I_7 I_8)_{O_3}$ and its size is 2. The candidate α is shown in Figure 3. The new pseudo-input is I_9 and the pseudo-output is O_5 . The modified D -matrix is given in Table 3.
6. Inputs I_7, I_8 and I_9 can be merged with I_1, I_2 and I_4 respectively. The number of original inputs merged (x) is 2.
7. Since $q = x$, exit with success.

The test mode version of the circuit in Figure 3 with the breakpoint inserted and merged inputs is shown in Fig 4. \square

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
O_1	1	1	1	1	1	1	0	0	0
O_2	0	1	1	1	1	1	0	0	0
O_3	1	1	1	0	0	1	0	0	1
O_4	0	0	1	1	1	1	1	1	0
O_5	0	0	0	0	0	0	1	1	0

Table 3: Modified dependency matrix

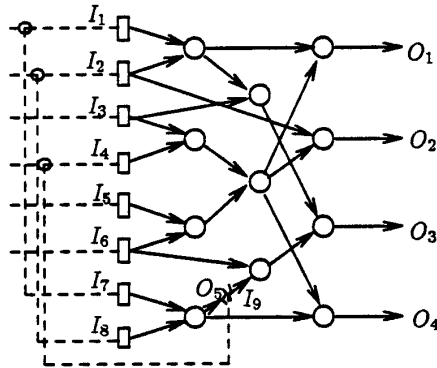


Figure 4: Graph model of modified example circuit

4 Experimental Results

The procedure *mtc* is implemented in *C++* and tested on a variety of circuits including the ISCAS-85 combinational benchmark circuits [10]. Table 4 presents some experimental results. Columns 1 and 2 refer to various circuits and their (n, m, k) characteristics. The number of inputs merged prior to determining breakpoints is given in column 3. After merging the unrelated inputs, the circuit is checked for maximal test concurrency. The results are noted in column 4. The total number of candidates for breakpoints available in the circuit is listed in column 5. The minimum number of breakpoints required to achieve maximal test concurrency and the number of solutions with this minimum breakpoints are given in column 6 and 7 respectively.

The circuit *mtc1* is the example circuit whose graph model is shown in Figure 3. Note that most of the benchmark circuits, namely the last four in the table, are MTC circuits. Execution was aborted for the benchmark circuit *c2670* due to memory limitations. The execution time is curtailed for the benchmark circuit *c3540*.

5 Conclusions

This paper presents a novel method for partitioning circuits such that they can be pseudo-exhaustively tested with a minimum number of test signals in a single test session. Though it is applied to circuits partitioned as output cones, the technique is applicable to other partitioning methods. A combined strategy is under investigation for partitioning circuits such that

1	2	3	4	5	6	7
ckt	(n,m,k)	merge	MTC	cand	bpt	soln
<i>mtc1</i>	(8,4,6)	0	no	17	1	6
<i>mtc2</i>	(10,5,7)	0	no	3	1	2
<i>mtc3</i>	(5,6,4)	0	no	24	3	91
<i>c60</i>	(15,4,10)	5	yes	11*	NA	-
<i>c880</i>	(60,26,45)	15	yes	264*	NA	-
<i>c1355</i>	(41,32,41)	0	yes	32*	NA	-
<i>c1908</i>	(33,25,33)	0	yes	0*	NA	-
<i>c2670</i>	(233,140,57)	-	-	-	-	-
<i>c3540</i>	(50,22,46)	1	no	32	>2	-

*Candidates are not needed since it is a MTC circuit. NA - not applicable.

Table 4: Results on benchmark circuits

the requirements - (a) the number of inputs to segments is restricted to some user-defined value, and (b) the circuit is maximal test concurrent in test mode - are met simultaneously rather than in a two-step procedure.

References

- [1] W. B. Jone and C. A. Papachristou. A Coordinated Approach to Partitioning and Test Pattern Generation for Pseudoexhaustive Testing. In *Proc. Design Automation Conf.*, pages 525-530, June 1989.
- [2] S. Hellebrand and H.-J. Wunderlich. Tools and Devices Supporting the Pseudo-Exhaustive Test. In *Proc. 1st European Design Automation Conf.*, March 1990.
- [3] E. J. McCluskey. Built-In Verification Test. In *Proc. Int'l Test Conf.*, pages 183-190, Sept 1982.
- [4] S. Hellebrand, H.-J. Wunderlich, and O. F. Haberl. Generating Pseudo-Exhaustive Vectors for External Testing. In *Proc. Int'l Test Conf.*, pages 670-679, Sept 1990.
- [5] D. T. Tang and L. S. Woo. Exhaustive Test Pattern Generation with Constant Weight Vectors. *IEEE Trans. on Computers*, C-32(12):1145-1150, December 1983.
- [6] J. G. Udell Jr. Reconfigurable Hardware for Pseudo-Exhaustive Test. In *Proc. Int'l Test Conf.*, pages 522-530, Sept 1988.
- [7] D. T. Tang and C. L. Chen. Logic Test Pattern Generation using Linear Codes. *IEEE Trans. on Computers*, C-33(9):845-850, September 1984.
- [8] L. T. Wang and E. J. McCluskey. Condensed Linear Feedback Shift Register (LFSR) Testing - A Pseudoexhaustive Test Technique. *IEEE Trans. on Computers*, C-35(4):367-370, April 1986.
- [9] Narsingh Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice Hall, Inc., EngleWood Cliffs, N.J., U.S.A., 1974.
- [10] F. Brglez and H. Fujiwara. A Neutral Netlist of Ten Combinational Benchmark Circuits and a Target Translator in FORTRAN. In *Proc. Int'l. Symp. on Circuits and Systems*, June 1985.