An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing

Rajagopalan Srinivasan, Sandeep K. Gupta
and Melvin A. Breuer

CENG Technical Report 93-08

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4469

An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing*

Rajagopalan Srinivasan, Sandeep K. Gupta and Melvin A. Breuer
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2563.

Abstract — Pseudo-exhaustive testing involves applying all possible input patterns to individual output cones of a circuit. Circuits with output cones driven by a large number of inputs often need to be partitioned to reduce the test application time. In this report the partitioning problem is first formulated as an integer linear programming (ILP) problem. To handle large circuits, we also present an efficient heuristic partitioning procedure of polynomial complexity. Experiments on combinational benchmark circuits validate the efficiency and quality of our approach.

Keywords: Pseudo-exhaustive testing, partitioning and test application time.

1 Introduction

Exhaustive testing of a combinational circuit involves exercising the circuit with all possible input patterns. It ensures detection of all detectable combinational faults in the circuit. A combinational fault does not manifest any sequential behavior of the circuit and is testable with a single input pattern. However, for circuits with large number of inputs, exhaustive testing is very time consuming and may not be practical.

To reduce test time without compromising on the coverage of the stuck-at faults, a circuit can be tested pseudo-exhaustively. Pseudo-exhaustive testing ensures detection of all detectable multiple stuck-at faults in the circuit. In pseudo-exhaustive testing, the circuit is partitioned into subcircuits, referred to as *segments*, and each segment is tested exhaustively.

*This work was supported by the Defense Advanced Research Projects Agency and monitored by the Federal Bureau of Investigation under Contract No. JFBI90092.

The segments need not be disjoint and each segment can have multiple outputs. The *size* of a segment is determined by the number of its inputs. Segments with single outputs are referred to as *output cones*. Pseudo-exhaustive testing also ensures detection of all detectable combinational faults within individual segments of the circuit.

The time required for pseudo-exhaustive testing depends on the sizes of the segments. Segment sizes can be reduced by placing segmentation cells in the circuit [1]. These cells are transparent in normal mode and act as pseudo-inputs and pseudo-outputs during test mode. It is desirable to place a minimum number of cells in the circuit so that the segments can be exhaustively tested within a prescribed time limit. This is referred to as the *partitioning problem for pseudo-exhaustive testing.*

Several researchers have attempted to solve different variations of this partitioning problem. McCluskey and Bozorgui-Nesbat [2] partitioned circuits with multiplexers to gain access to the segments. Their method deals with a block level rather than gate level model of a circuit leading to high hardware overhead. Roberts and Lala [3] considered a localized strategy requiring large number of segmentation cells. Bhatt et al. [4] considered partitioning with the following design constraints: (1) all paths from inputs to outputs should encounter the same number of segmentation cells; and (2) levelized circuits in which segmentation cells are placed on all gates at a level or on none at all. Jone and Papachristou [5] developed a coordinated approach for partitioning and exhaustive test pattern generation of segments. It ensures disjoint segments and feasible test pattern generators but results in a large number of segmentation cells. Hellebrand and Wunderlich [1] proposed the hill-climbing procedures with optional backtracking resulting in locally optimized solutions.

We have formulated the partitioning problem as the classical integer linear programming (ILP) problem. The *objective* is to minimize the number of segmentation cells to be placed in the circuit, subject to the *constraints* that in the test mode all cones should depend on no more than a specified number of inputs. For small circuits, optimal solutions can be obtained by solving the ILP formulation. However, the formulation may not be computationally viable for large circuits since the number of constraints grows non-linearly with the number of levels in the circuit. We have developed an efficient heuristic procedure applicable to large circuits. The procedure is of polynomial complexity and is based on the concept of articulation points [6]. The results on the ISCAS combinational benchmark circuits [7] and comparison with the results of others validates the efficiency of our heuristic approach.

The report is organized as follows. Section 2 presents the partitioning strategy and the problem formulation. The heuristic procedure is described in section 3. The experimental results and the conclusions are provided in sections 4 and 5, respectively.

2 Cone Size Reduction

Consider a combinational circuit with n inputs and m outputs. An output is said to *depend* on an input if there exists at least one path from that input to the output. The set of inputs on which an output depends is referred to as the *dependency set* of that output. The *dependency* for an output is given by the cardinality of its dependency set. Assume each output has a dependency of k or less inputs and at least one output has the dependency of exactly k inputs. The circuit can be characterized as an (n, m, k) circuit. Exhaustive testing of the circuit requires 2^n patterns which may be prohibitive for large values of n.

For pseudo-exhaustive testing, we shall consider segments that form the output cones. The (n, m, k) circuit can be divided into m output cones and each cone can be exhaustively tested with 2^k patterns. However, a minimum of 2^k patterns is required to exhaustively test all the cones in the circuit. Some cones may not be tested simultaneously with 2^k patterns due to conflicts in their input requirements and hence the circuit may require more than 2^k patterns for pseudo-exhaustive testing. If k is sufficiently smaller than n, the time taken for pseudo-exhaustive testing can be much less compared to exhaustive testing.

For some circuits, k can be large enough to prohibit the testing even with the least required 2^k patterns. To reduce test time, the size of any output cone can be restricted to some predetermined value (say r, where $r < k$). This is referred to as *cone size reduction* and can be achieved by partitioning the circuit by placing few segmentation cells (say s). In the test mode, the original (n, m, k) circuit gets modified to an $(n + s, m + s, r)$ circuit.

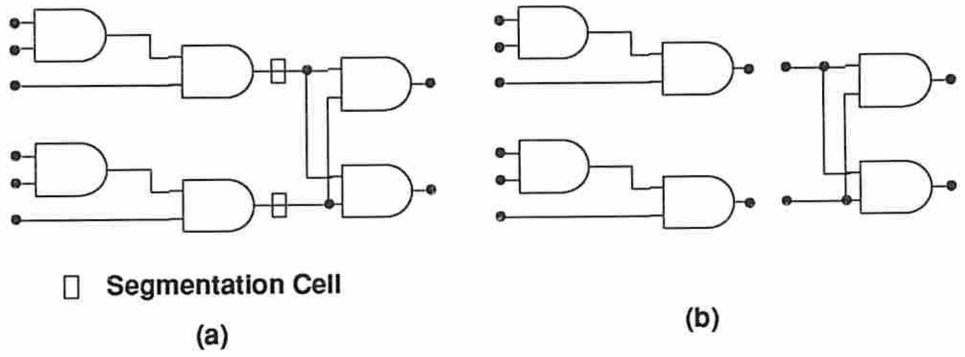

Figure 1: A partitioned circuit in (a) normal mode (b) test mode

Example 1 Consider the $(6, 2, 6)$ circuit shown in Figure 1(a). The circuit requires 2^6 patterns for both exhaustive testing and pseudo-exhaustive testing (without partitioning). The circuit can be partitioned with two segmentation cells such that each cone depends on at most three inputs. In the test mode, the circuit becomes an $(8,4,3)$ circuit shown in Figure 1(b). All the cones can be exhaustively tested concurrently with 2^3 patterns. □

2.1 Partitioning Strategy

Our goal is to reduce the sizes of output cones of the circuit by placing segmentation cells so that every cone has an acceptable number of inputs. Segmentation cells add area overhead to the original circuit and introduce delays during normal mode of operation. Hence it is desirable to have a minimum number of these cells to achieve our goal. We shall describe two partitioning strategies for cone size reduction.

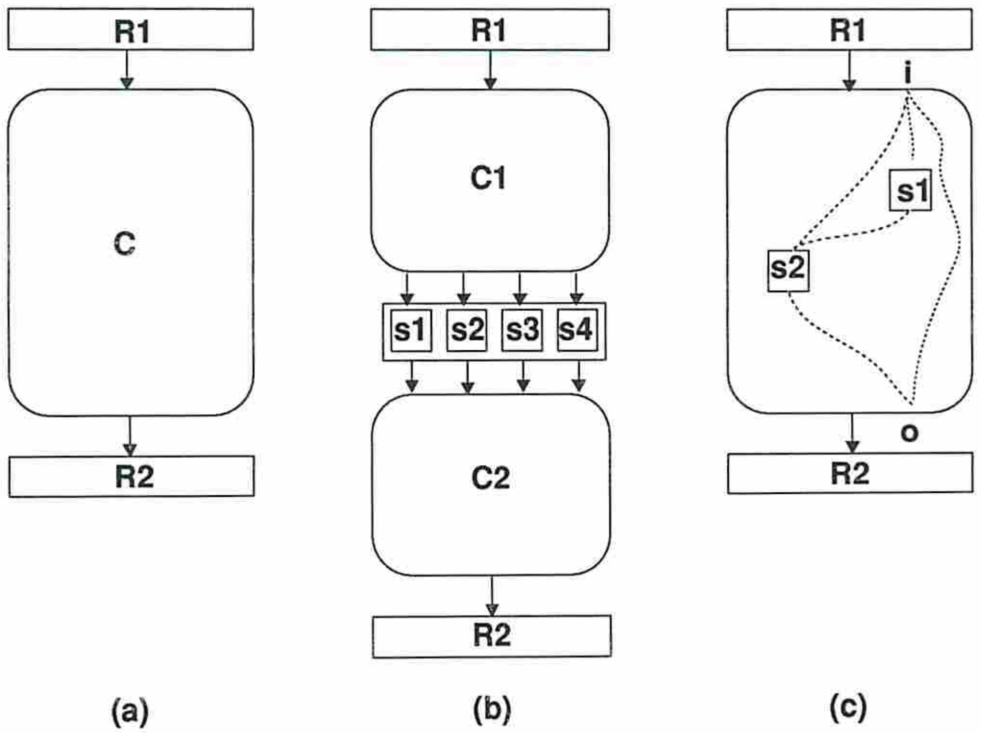

Figure 2: Partitioning strategies for cone size reduction (a) Unpartitioned circuit (b) Constrained partitioning strategy (c) Unconstrained partitioning strategy

Consider a combinational circuit C shown in Figure 2(a). The circuit is fed by register $R1$ and feeds register $R2$. Assume that some of its output cones have an unacceptable number of inputs. One strategy is to constrain the segments to be totally disjoint. The *constrained partitioning strategy* (CPS) is shown in Figure 2(b). The original circuit C is partitioned into two disjoint segments $C1$ and $C2$ such that all the output cones in both the segments have an acceptable number of inputs. Four segmentation cells $s1$ through $s4$ are placed to derive the segments. The problem of finding an optimal solution for CPS has been shown to be NP-complete and a heuristic procedure is given in [5]. CPS usually uses more than a minimum number of cells due to the constraint on segments. Also critical paths invariably gets affected since any path between an input and an output passes through at least one cell.

Another strategy is not to constrain the segments. The segments need not necessarily be disjoint. The *unconstrained partitioning strategy* (UPS) is shown in Figure 2(c). In this case, only two cells $s1$ and $s2$ may be required to achieve the goal. The original circuit C is not partitioned into disjoint segments. UPS forms a generalized case of CPS and hence yields better solutions. Critical paths can be left undisturbed in UPS, though in some cases extra cells may be required. It is shown in [4] that determining optimal solutions to UPS is NP-complete. Hill climbing procedures with backtracking is proposed in [1]. Our work is focused on achieving good sub-optimal solutions for UPS.

Under UPS, two interesting problems arise depending on where the segmentation cells are placed in the circuit. In the first case, cells can be placed only on the fanout stem of gate outputs (nodes), while in the second case, cells can be placed on all signal lines (edges). For the latter case, different cells placed on fanout signals of a gate output can be merged into a single cell. These are referred to as *node partitioning problem* and *edge partitioning problem* respectively. The formal statements of the problems are given below.

2.2 Problem Statement

Node partitioning problem (NPP): Given an (n, m, k) circuit with f being the maximum fanin of any gate and integer r such that $f \leq r < k$, find an assignment of minimum number of segmentation cells to *fanout stems of gate outputs (nodes)* such that all the segments (output cones) depend on a maximum of r inputs.

Edge partitioning problem (EPP): Given an (n, m, k) circuit with f being the maximum fanin of any gate and integer r such that $f \leq r < k$, find an assignment of minimum number of segmentation cells to *fanout stems and/or branches of gate outputs (edges)* such that all the segments (output cones) depend on a maximum of r inputs.

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

Example 2 Consider the graph model of a $(4, 3, 4)$ circuit shown in Figure 3. The outputs x, y and z depends on the input sets $\{a, b, c\}$, $\{a, b, c, d\}$ and $\{b, c, d\}$ respectively. Suppose

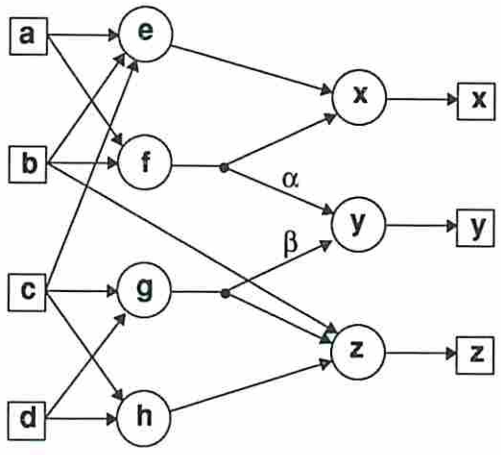

Figure 3: NPP and EPP

we wish to restrict the output cone sizes to three inputs. The two best solutions for NPP require two cells to be placed on (1) the fanout stems of gates f and e; or (2) the fanout stems of gates g and h. However for the EPP, two best solutions can be obtained by placing only one cell on either signal α or signal β. The two cases can be explained as follows.

First let us consider the NPP. Since only output y depends on four inputs, it requires a cell to be placed on one of the gates f or g. However placing a cell on the output of either of these gates increases the input dependency on other outputs. Therefore placing a cell on gate f calls for an additional cell to be placed on gate e and placing a cell on gate g calls for an additional cell on gate h. Next consider the EPP. To restrict the input dependency on output y, a cell need to be placed on either of the signal α or β. Placing a cell on any of these signal lines does not affect the input dependency of other outputs. □

2.3 Problem Formulation

Both NPP and EPP are formulated as classical integer linear programming (ILP) problems. The *objective function* is to minimize the number of segmentation cells to be placed in the circuit. The *constraints* are that all gates depend on less than or equal to r inputs in test mode. The value of r is user-specified and dictates the size of any output cone of the circuit in test mode. The problem formulation is illustrated using the graph shown in Figure 5 which forms the graph model of the circuit in Figure 1(a).

Notation:

$(\vee)$ :: logical OR operation

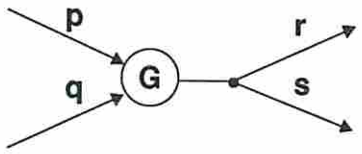

Figure 4: Example gate

$$
\begin{aligned}
(\wedge) &\;::\; \text{logical AND operation} \\
(\cup) &\;::\; \text{set union operation} \\
v_i &\;::\; \text{pseudo-input created when a cell is placed on the } i\text{th signal} \\
S_i &\;::\; \text{dependency set for signal } i \\
W_i &= |S_i|\ ;\ \text{dependency value for signal } i \\
x_i &= \begin{cases} 1 & \text{if a cell is placed on the } i\text{th signal,} \\ 0 & \text{otherwise.} \end{cases} \\
S_i x_i &= \begin{cases} S_i & \text{if } x_i \text{ is 1,} \\ \emptyset & \text{otherwise.} \end{cases} \\
|S_i x_i| &= \begin{cases} |S_i| & \text{if } x_i \text{ is 1,} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}
$$

For a logic gate in the circuit, we can compute the dependency sets of the fanouts from the dependency sets of the fanins. Consider the gate G shown in Figure 4 with fanins and fanouts as $\{p, q\}$ and $\{r, s\}$ respectively. The set of inputs feeding G through the fanin p is either S_p or $\{v_p\}$ depending on whether a segmentation cell is placed on p. The set of inputs feeding G through the fanin p can be expressed as $S_p\bar{x}_p \cup \{v_p\}x_p$. The expression is valid because of the complementary nature of the boolean variables x_p and $\bar{x}_p$. The number of inputs feeding G through the fanin p is $|S_p\bar{x}_p \cup \{v_p\}x_p| = |S_p|\bar{x}_p + x_p$. Similarly, the set of inputs feeding G through the fanin q can be expressed as $S_q\bar{x}_q \cup \{v_q\}x_q$. The number of inputs feeding G through the fanin q is $|S_q\bar{x}_q \cup \{v_q\}x_q| = |S_q|\bar{x}_q + x_q$. The dependency sets S_r and S_s are given by

$$
\begin{aligned}
S_r = S_s &= \{S_p\bar{x}_p \cup \{v_p\}x_p\} \cup \{S_q\bar{x}_q \cup \{v_q\}x_q\} \\
&= \{S_p \cup S_q\}\bar{x}_p\bar{x}_q \cup \{\{v_p\} \cup S_q\}x_p\bar{x}_q \cup \{S_p \cup \{v_q\}\}\bar{x}_p x_q \cup \{v_p, v_q\}x_p x_q
\end{aligned} \tag{2}
$$

and the dependency values W_r and W_s are given by

$$
\begin{aligned}
W_r = W_s = |S_r| &= |S_s| \\
&= |S_p \cup S_q|\bar{x}_p\bar{x}_q + |\{\{v_p\} \cup S_q\}|x_p\bar{x}_q + |\{S_p \cup \{v_q\}\}|\bar{x}_p x_q + 2x_p x_q
\end{aligned} \tag{3}
$$

In Equation 3, if one of the product terms is satisfied, the rest of the terms becomes zero. In other words the terms are mutually exclusive. Hence the addition of the terms is valid. Cells placed on different fanout branches of a gate can be merged together and replaced with a single cell. For example, cells placed on signals r and s can be combined to a single cell and placed on the fanout stem.

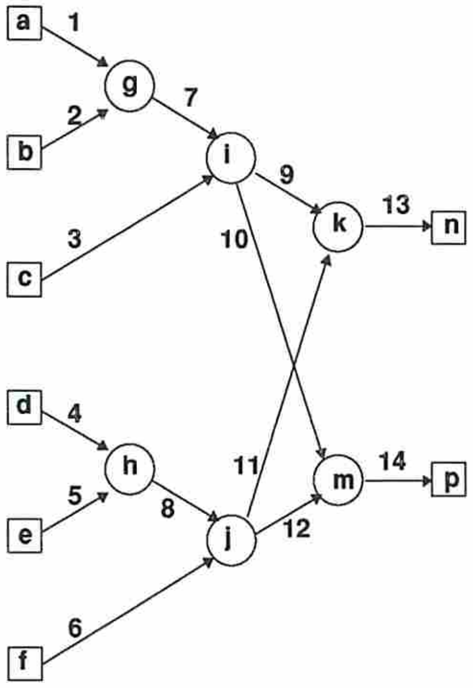

Figure 5: Circuit graph

Edge Partitioning Problem (EPP): Consider the graph model in Figure 5. The primary inputs are labelled a through f and the signals are numbered 1 through 14. The objective is to minimize the function

$$F = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + (x_9 \vee x_{10}) + (x_{11} \vee x_{12}) + x_{13} + x_{14} \quad (4)$$

subject to the constraints $W_i \leq r, \ \forall i = 1, \ldots, 14$. Let the cone size limit (r) be 3.

The non-linearity of the objective function is due to the following reason. For the fanouts of the gate i, if cells are placed on signals 9 and 10, they can be merged as a single cell and hence they appear as $x_9 \vee x_{10}$ in the objective function. The pseudo-inputs created for the fanouts, v_9 and v_{10}, are the same since the cells are merged.

Placing cells on the primary inputs and primary outputs do not contribute to the solution. Hence $x_1 = \ldots = x_6 = 0$ and $x_{13} = x_{14} = 0$. Therefore the objective function in Equation 4 is reduced to

$$F = x_7 + x_8 + (x_9 \vee x_{10}) + (x_{11} \vee x_{12}) \quad (5)$$

Linearization: Non-linear programming arises due to non-linear terms in either the objective function or constraints. The non-linear terms can be avoided by the following transformation steps.

1. Replace each nonlinear term by a new boolean variable. An *OR* term of the form $x_1 \vee x_2 \vee \ldots \vee x_n$ is replaced by a variable y and an *AND* term $x_1 \wedge x_2 \wedge \ldots \wedge x_n$ is replaced by z.

2. Add two constraints for each non-linear term. For the *OR* term, add the constraints $y \leq x_1 + x_2 + \ldots + x_n \leq ny$. For the *AND* term, add the constraints $nz \leq x_1 + x_2 + \ldots + x_n \leq z + n - 1$.

It can be easily verified that the new variables and the constraints preserve the original problem. Hence the non-linear objective function of Equation 5 can be replaced by the new linear objective function

$$F = x_7 + x_8 + y_1 + y_2 \tag{6}$$

along with four new constraints

$$\begin{aligned} y_1 &\leq x_9 + x_{10} \leq 2y_1 \\ y_2 &\leq x_{11} + x_{12} \leq 2y_2 \end{aligned} \tag{7}$$

The dependency sets S_i and the dependency values W_i, $\forall\ i = 1, \ldots, 14$ are computed and listed below.

$$\begin{aligned}
S_1 &= \{a\};\ S_2 = \{b\};\ S_3 = \{c\} \\
S_4 &= \{d\};\ S_5 = \{e\};\ S_6 = \{f\} \\
S_7 &= S_1 \cup S_2 = \{a, b\};\quad W_7 = 2 \\
S_8 &= S_4 \cup S_5 = \{d, e\};\quad W_8 = 2 \\
S_9 &= S_{10} = \{S_7\bar{x}_7 \cup \{v_7\}x_7\} \cup \{S_3\} \\
&= \{\{a, b\}\bar{x}_7 \cup \{v_7\}x_7\} \cup \{c\} \\
&= \{a, b, c\}\bar{x}_7 \cup \{v_7, c\}x_7 \\
W_9 &= W_{10} = |S_9| = |S_{10}| \\
&= 3\bar{x}_7 + 2x_7 \\
S_{11} &= S_{12} = \{S_8\bar{x}_8 \cup \{v_8\}x_8\} \cup \{S_6\} \\
&= \{\{d, e\}\bar{x}_8 \cup \{v_8\}x_8\} \cup \{f\} \\
&= \{d, e, f\}\bar{x}_8 \cup \{v_8, f\}x_8 \\
W_{11} &= W_{12} = |S_{11}| = |S_{12}|
\end{aligned}$$

$$
\begin{aligned}
&= 3\bar{x}_8 + 2x_8 \\
S_{13} &= \{S_9\bar{x}_9 \cup \{v_9\}x_9\} \cup \{S_{11}\bar{x}_{11} \cup \{v_{11}\}x_{11}\} \\
&= \{S_9 \cup S_{11}\}\bar{x}_9\bar{x}_{11} \cup \{\{v_9\} \cup S_{11}\}x_9\bar{x}_{11} \cup \\
&\quad \{S_9 \cup \{v_{11}\}\}\bar{x}_9 x_{11} \cup \{v_9, v_{11}\}x_9 x_{11} \\
&= (\{a,b,c,d,e,f\}\bar{x}_7\bar{x}_8 \cup \{a,b,c,v_8,f\}\bar{x}_7 x_8 \cup \\
&\quad \{v_7,c,d,e,f\}x_7\bar{x}_8 \cup \{v_7,c,v_8,f\}x_7x_8)\bar{x}_9\bar{x}_{11} \cup \\
&\quad (\{d,e,f,v_9\}\bar{x}_8 \cup \{v_8,v_9,f\}x_8)x_9\bar{x}_{11} \cup \\
&\quad (\{a,b,c,v_{11}\}\bar{x}_7 \cup \{v_7,c,v_{11}\}x_7)\bar{x}_9 x_{11} \cup \\
&\quad \{v_9,v_{11}\}x_9x_{11} \\
W_{13} &= (6\bar{x}_7\bar{x}_8 + 5\bar{x}_7x_8 + 5x_7\bar{x}_8 + 4x_7x_8)\bar{x}_9\bar{x}_{11} + \\
&\quad (4\bar{x}_8 + 3x_8)x_9\bar{x}_{11} + (4\bar{x}_7 + 3x_7)\bar{x}_9x_{11} + 2x_9x_{11} \\
S_{14} &= \{S_{10}\bar{x}_{10} \cup \{v_{10}\}x_{10}\} \cup \{S_{12}\bar{x}_{12} \cup \{v_{12}\}x_{12}\} \\
&= \{S_{10} \cup S_{12}\}\bar{x}_{10}\bar{x}_{12} \cup \{\{v_{10}\} \cup S_{12}\}x_{10}\bar{x}_{12} \cup \\
&\quad \{S_{10} \cup \{v_{12}\}\}\bar{x}_{10}x_{12} \cup \{v_{10},v_{12}\}x_{10}x_{12} \\
&= (\{a,b,c,d,e,f\}\bar{x}_7\bar{x}_8 \cup \{a,b,c,v_8,f\}\bar{x}_7x_8 \cup \\
&\quad \{v_7,c,d,e,f\}x_7\bar{x}_8 \cup \{v_7,c,v_8,f\}x_7x_8)\bar{x}_{10}\bar{x}_{12} \cup \\
&\quad (\{d,e,f,v_{10}\}\bar{x}_8 \cup \{v_8,v_{10},f\}x_8)x_{10}\bar{x}_{12} \cup \\
&\quad (\{a,b,c,v_{12}\}\bar{x}_7 \cup \{v_7,c,v_{12}\}x_7)\bar{x}_{10}x_{12} \cup \\
&\quad \{v_{10},v_{12}\}x_{10}x_{12} \\
W_{14} &= (6\bar{x}_7\bar{x}_8 + 5\bar{x}_7x_8 + 5x_7\bar{x}_8 + 4x_7x_8)\bar{x}_{10}\bar{x}_{12} + \\
&\quad (4\bar{x}_8 + 3x_8)x_{10}\bar{x}_{12} + (4\bar{x}_7 + 3x_7)\bar{x}_{10}x_{12} + 2x_{10}x_{12}
\end{aligned}
$$

Since none of the signals should depend on more than three inputs, from W_{13} we have the constraints

$$\bar{x}_9\bar{x}_{11} = 0; \quad \bar{x}_8x_9\bar{x}_{11} = 0; \quad \bar{x}_7\bar{x}_9x_{11} = 0 \tag{8}$$

Similarly, from W_{14} we have the constraints

$$\bar{x}_{10}\bar{x}_{12} = 0; \quad \bar{x}_7\bar{x}_{10}x_{12} = 0; \quad \bar{x}_8x_{10}\bar{x}_{12} = 0 \tag{9}$$

The constraints in Equations 8 and 9 are linear since we can interpret them as their duals. For example, consider the first constraint of Equation 8. The dual of the constraint $\bar{x}_9\bar{x}_{11} = 0$ is $x_9 + x_{11} \geq 1$ which is linear. This constraint can be interpreted as requiring at least one cell to be placed on the signals among the set $\{9, 11\}$. If none is placed, then signal 13 will be fed by more than three inputs.

An optimal solution to the ILP problem with the objective function given by Equation 6 with the set of linear constraints given by Equations 7, 8 and 9 is given by

$$\begin{aligned} x_7 &= x_8 = 0; \qquad x_9 = x_{10} = x_{11} = x_{12} = 1 \\ F &= 2 \end{aligned}$$

Node Partitioning Problem (NPP): For the NPP, cells are allowed to be placed only on the fanout stem of gate outputs. The variables associated with the fanout signals of a gate are considered equivalent. Hence the variables $x_9 = x_{10}$ and $x_{11} = x_{12}$. The objective function given by Equation 5 gets linearized for the NPP and is given by

$$F = x_7 + x_8 + x_9 + x_{11} \tag{10}$$

and the constraints given by Equation 9 reduces to those of given by Equation 8. Thus for NPP, the objective function is linear with a reduced set of linear constraints. An optimal solution to the ILP problem with the objective function given by Equation 10 with the set of linear constraints given by Equation 8 is given by

$$\begin{aligned} x_7 &= x_8 = 0; \qquad x_9 = x_{11} = 1 \\ F &= 2 \end{aligned}$$

Thus both EPP and NPP have identical optimal solutions. The solution implies that two segmentation cells need to be placed at the fanout stems of gates i and j.

Complexity: The complexity of NPP depends on the number of gates, while that of EPP depends on the number of signals in the circuit. Since the number of gates is less compared to the number of signals, NPP involves smaller search space than EPP. But EPP forms a generalized case of NPP and hence yields better solutions than NPP.

Placing cells on certain nodes/edges (trivial cases include the primary inputs and outputs) do not contribute to the solution. Hence these nodes/edges need not be considered in the problem formulation. We will show in the next section that few more nodes/edges are gauranteed to be absent in any optimal solution. Thus the number of nodes/edges considered for the ILP formulation can be reduced to some extent. In spite of this reduction, the ILP formulation may still not be feasible for large circuits. This is due to the fact that the number of constraints grow non-linearly with the levels in the circuit. Thus the formulation can be used to obtain optimal solutions only for reasonably small circuits.

3 Heuristic Approach

To handle large circuits we have developed an efficient heuristic procedure based on the concept of articulation points [6]. The procedure is applicable to NPP and can be easily extended for EPP. We shall first present a few definitions that are illustrated using the circuit graph shown in Figure 5.

3.1 Definitions

Node x is said to *feed* node y (node y is said to be *fed by* node x) if there exists a directed path from node x to node y. The set of all nodes that feed a node is called its *fanin cone* and the set of all nodes that are fed by the node is called its *fanout cone*. The fanin and fanout cones of node i are $\{a, b, c, g\}$ and $\{k, m, n, p\}$, respectively. The set of all input nodes that feed a node is called its *dependency set*. The cardinality of the set gives the *dependency* of the node. The dependency set of node i is $\{a, b, c\}$ and its dependency is three.

An *articulation node* is a node that is contained in all paths originating from its dependency set and ending in its fanout cone. Node i is an articulation node since it is contained in all paths originating from its dependency set $\{a, b, c\}$ and ending in its fanout cone $\{k, m, n, p\}$. A circuit graph with no reconvergence among paths contains only articulation nodes.

Consider a node x and a node y in the fanout cone of node x. There may exist some input nodes that feed node y only via node x. The number of such input nodes is referred to as the *articulation value* of node x with respect to node y. Placing a segmentation cell after node x will decrease the dependency of node y by the articulation value. For example, the articulation value of node i with respect to node k is three since the input nodes a, b and c have all the paths from them to node k only via node i. The articulation value of an articulation node with respect to any node in its fanout cone is equal to its dependency. Since node i is an articulation node, its articulation value is equal to its dependency.

For a specified cone size limit, the set of all nodes that have their dependencies not greater than the limit is called the *good set*. The set of remaining nodes that have their dependencies greater than the limit is called the *bad set*. For the cone size limit of three, the good and the bad sets for the circuit graph are $\{a, b, c, d, e, f, g, h, i, j\}$ and $\{k, m, n, p\}$, respectively. The set of all nodes in the bad set that are fed directly by at least one node in the good set is called the *envelope*. For our example, the set $\{k, m\}$ forms the envelope.

Consider a node x in the good set such that it is not an input node and feeds at least one node in the bad set. The node x is called a *candidate node* if there exists no other node y in the good set such that all paths from x to bad set nodes go only via node y. For example, node g is not a candidate node since all paths from g to bad set nodes go only via the good set node i. On the contrary, nodes i and j are candidate nodes.

3.2 NPP and EPP

With the above definitions we shall present the salient characteristics of NPP and EPP. It will also be shown that it is sufficient to consider only the candidate nodes among the nodes in the good set for the placement of segmentation cells.

Lemma 1 *For circuits with reconvergent fanouts, EPP gives better results than NPP.*

Proof : Let us consider a circuit with reconvergent fanouts and an optimal solution for NPP. This forms a solution for EPP since the cells are placed on the fanout stems of nodes. An optimal solution for EPP can only have less than or at most the same number of cells than an optimal solution for NPP. It should be noted that the NPP is a special case of EPP. □

Lemma 2 *For circuits without reconvergent fanouts, both NPP and EPP give equally good results.*

Proof : For a circuit without reconvergent fanouts, the fanout branches of any node always feed different output cones. Consider any node in the circuit that has fanout greater than one. Placing a cell on one of the fanout branches feeding some output cone is advantageous to that output cone. However placing the same cell on the fanout stem instead of the fanout branch will still be equally advantageous for that output cone. This cell will never increase the dependencies of other output cones fed by that node. Hence it is always better to place the cell on the fanout stem rather than on its fanout branches. Therefore both NPP and EPP have the same search space giving rise to equally good results. □

Hence, for circuits without reconvergent fanouts, it is sufficient to consider NPP. Any optimal solution for NPP can be transformed to corresponding optimal solution for EPP and vice-versa. However, if the circuit has reconvergent fanouts, the two-way transformation is not feasible as evident by Example 2.

Lemma 3 *Suppose there exists an articulation node with dependency no greater than the cone size limit. Then the fanout stems and branches of the nodes feeding the articulation node need not be considered for placing segmentation cells.*

Proof : Let node x be an articulation node with dependency not greater than the cone size limit. Consider node y in the fanin cone of node x. Let the dependency sets for nodes x and y be S_x and S_y, respectively. Placing a segmentation cell on the fanout stem of the articulation node x will reduce the dependency by $|S_x|$ for all the nodes in its fanout cone. However, placing a cell on the fanout stem or branches of node y can only reduce the dependency up to a maximum of $|S_y|$. Since $S_y \subseteq S_x$, it is better to place a cell after node x than after node y. Hence, it is equally good if not better to place a cell on the articulation node x than to place a cell anywhere in its fanin cone. □

Since the graph in Figure 5 has no reconvergent fanouts, Lemma 2(a) is applicable and therefore NPP can be considered instead of EPP. Both EPP and NPP have the same objective function given by Equation 10, the same set of constraints given by Equation 8, and have identical solutions. It should be noted that all the nodes in the graph are articulation nodes. Hence Lemma 3 can also be applied to derive the optimal solution. Since nodes i and j are articulation nodes with dependencies not greater than the cone size limit, nodes g and h need not be considered for placing cells. Hence substituting $x_7 = x_8 = 0$ in the objective function given by Equation 10 and in the constraints given by Equation 8 again leads to the same optimal solution.

Lemma 4
(a) For circuits without fanouts, optimal solutions for both NPP and EPP can be determined in polynomial time.
(b) For circuits with fanouts, determining optimal solutions for both NPP and EPP are NP-complete.

Proof : (a) A circuit without fanouts has only one output and its graph model forms a tree structure. All the nodes in the circuit graph form articulation nodes. By Lemma 2, both NPP and EPP lead to the same optimal solution. Starting from the inputs, the dependency of each node can be determined in a breadth first manner. Let the dependency of node i exceed the specified limit. Since node i is an articulation node, by Lemma 3, only the inputs to node i are considered for placing segmentation cells. A cell is placed on one of its inputs so that the dependency of node i gets reduced by the maximum extent. If necessary, more cells are placed on other inputs to node i in order to reduce its dependency below the limit. The procedure is repeated for all nodes until the output is reached. This process is linear in the number of nodes in the circuit. It results in an optimal solution since the cells are placed only at indispensible nodes.

(b) For circuits with fanouts, it has been proven in [4] that NPP is *NP-complete.* EPP belongs to NP as its solution can be verified in linear time. Since NPP forms a special case of EPP, EPP is also *NP-complete.* □

Theorem 1 *For all circuits, it is sufficient to consider only the candidate nodes among all nodes in the good set for placing segmentation cells.*

Proof : Placing a segmentation cell after either an input node or a node that does not feed any node in the bad set has no benefits. Consider a node x in the good set that feeds at least one node in the bad set and is not a candidate node. Then there exists a candidate node y such that all paths from node x to nodes in the bad set go only via node y. The articulation value of node x with respect to any node z in the bad set is always less than or equal to that of the articulation value of node y with respect to node z. Therefore, the benefit of placing a segmentation cell after node y is always better than placing after node x. Hence it is sufficient to consider only candidate nodes among all nodes in the good set for placing the segmentation cells. □

3.3 Heuristics

The heuristics are based on the articulation values of the nodes in the circuit. Placing a segmentation cell after a node benefits the nodes in its fanout cone. For a cell placed after node x, the dependency of node y in its fanout cone is decreased by an amount equal to the articulation value of node x with respect to node y. Thus the benefit of placing a cell after a node is measured in terms of its articulation value with respect to other nodes in its fanout cone. However the cell manifests as a pseudo-input and that should be accounted for in measuring the benefits.

Placing a cell after a node in the bad set requires the future placement of cells in its fanin cone to reduce its own dependency. Due to this nature, the measured benefits for the cells placed earlier in the process tend to get drastically reduced. On the contrary, placing a cell after a node in the good set does not require placement of another cell in its fanin cone. Hence we only consider the nodes in the good set for placing segmentation cells. Among these nodes, as per Theorem 1, it is sufficient to consider only the candidate nodes.

There are three minor variations to the main theme in our heuristic approach. These variations form the alternate heuristics H1, H2 and H3. The candidates are identified among the nodes in the good set. The benefit of placing a segmentation cell after a candidate is quantified by the heuristic measure. The measure is primarily based on its articulation value with respect to the nodes in its fanout cone and varies with the heuristics.

- **Heuristic H1** gives equal importance to the candidates and nodes in the bad set. The measure for a candidate is the cumulative sum of its articulation value with respect to all nodes in the bad set and other candidates.
- **Heuristic H2** gives twice the importance to the nodes in the bad set than to other candidates. Since the candidates already have their dependencies less than r, the

benefits for the nodes in the bad set are stressed compared to the benefits for the candidates.

- **Heuristic H3** measures the benefits for the nodes in the envelope. Since the nodes in the envelope feed the rest of the nodes in the bad set, the benefits for the nodes in the bad set are implicitly considered.

3.4 Procedure

We shall now provide the details of the heuristic procedure. Among the three heuristics H1, H2 and H3, one of them is selected prior to invoking the procedure.

Procedure CSR

Input: Circuit graph with cone size limit r.
Output: Partitioned graph with reduced cone sizes.

1. Determine the good and bad sets, envelope and the candidates.
2. For each candidate i, apply the selected heuristics to obtain its heuristic measure (H_i).
    - *(H1:)* $H_i = \Sigma$ articulation value of node i with respect to all bad set nodes + Σ articulation value of node i with respect to other candidates.
    - *(H2:)* $H_i = 2 \times \Sigma$ articulation value of node i with respect to all bad set nodes + Σ articulation value of node i with respect to other candidates.
    - *(H3:)* $H_i = \Sigma$ articulation value of node i with respect to all envelope nodes.
3. Select the candidate with the highest heuristic measure and place a segmentation cell after the node.
4. Update the dependencies of the nodes. If there exists a node with dependency greater than r, go to step 1.
5. Examine the segmentation cells and remove unnecessary ones.

The iterative procedure progressively adds nodes to the good set from the bad set. The procedure terminates when the bad set becomes empty. Due to the greedy nature of the procedure, cells placed during some iterations may not be necessary at the end of the process. In the final step, the cells are examined and unnecessary ones are removed from the circuit.

The main step is determining the heuristic measure for the candidates. The time complexity for determining the articulation value of a candidate with respect to the nodes in its fanout cone is $O(N)$, where N is the number of nodes in the circuit. Repeating for all the candidates makes the complexity of the procedure $O(N^2)$.

4 Experimental Results

Procedure CSR is implemented in C++ as part of our object-oriented database system [8]. Experiments were conducted on several circuits including eight ISCAS combinational benckmark circuits [7]. The results on the benchmark circuits are compared with those in [1].

Table 1: Results on benchmark circuits

		r = 20					r = 16				
Ckt	(n,m,k)	HW1	HW2	H1	H2	H3	HW1	HW2	H1	H2	H3
c432	(36,7,36)	20*	21	20*	20*	20*	27*	27*	27*	27*	27*
c499	(41,32,41)	9	9	8*	8*	8*	8*	8*	8*	8*	8*
c880	(60,26,45)	14	14	11	10*	11	19	16	11*	11*	11*
c1355	(41,32,41)	9	9	8*	8*	8*	8*	8*	8*	8*	8*
c1908	(33,25,33)	18	17	14*	14*	15	21	22	18*	18*	18*
c2670	(233,140,120)	37	29*	29*	29*	29*	45	33*	33*	33*	33*
c3540	(50,22,50)	63	68	60*	68	61	87*	90	91	93	87*
c5315	(178,123,67)	42	46	37*	37*	38	52*	62	54	52*	54

Table 1 presents the experimental results on the benchmark circuits. Column two describes the characteristics of the circuits. For example, circuit *c*432 has 36 inputs, 7 outputs with a maximum dependency of 36 inputs. Two sets of experiments were conducted for restricting the dependencies to 20 inputs and 16 inputs, respectively. The subsequent columns denote the number of segmentation cells required for partitioning the circuits to satisfy the specified cone size restriction value (r). HW1 and HW2 are the heuristics proposed in [1] and H1, H2 and H3 are the heuristics proposed in this report. For example, for cone size reduction value (r) equal to 20 inputs, circuit *c*432 requires 20, 21, 20, 20 and 20 segmentation cells using the heuristics HW1, HW2, H1, H2 and H3 respectively. Entries with asterisks indicate the best results obtained by the five procedures.

Heuristics HW1 and HW2 are basically hill-climbing procedures with optional backtracking in the search process. Among the nodes in the circuit graph, the first node (say node v) exceeding the cone restriction value (r) is identified. Heuristic HW1 uses an exponential complexity routine to determine an optimal number of cells for placing in the fanin cone of node v to restrict its dependency. The procedure is repeated till all the output cones have acceptable dependencies. The technique tends to neglect the global effects on placing the individual cells. Heuristic HW2 considers the global effects but cells are allowed to be placed only on certain nodes in the fanin cone of node v. The heuristic measure adopted for both HW1 and HW2 may not be very appropriate in guiding the search space.

Our heuristics consider all the bad set nodes simultaneously. We consider a complete set of candidate nodes in comparision to the candidate nodes considered in HW1 and HW2. Our heuristic measures evaluate the placement of cells on candidate nodes with respect to all the bad set nodes. This evaluation provides a global approach to the problem and results in good sub-optimal solutions. For all the circuits considered, the best result produced by our heuristics is always better than or equal to the best results produced by heuristics HW1 and HW2.

Table 2: Segmentation cells required for various cone sizes

Ckt	(n,m,k)	r	40	36	32	28	24	20	16	12
c432	(36,7,36)	s	-	-	9	13	16	20	27	35
		k'	-	-	32	28	24	20	16	12
c499	(41,32,41)	s	2	6	6	7	7	8	8	16
		k'	39	32	32	22	22	14	14	10
c880	(60,26,45)	s	1	2	4	9	9	11	11	19
		k'	37	36	32	24	24	20	16	12
add16	(33,17,33)	s	-	-	1	1	1	1	2	3
		k'	-	-	31	27	23	19	15	11
mult8	(18,16,18)	s	-	-	-	-	-	-	6	22
		k'	-	-	-	-	-	-	16	12
ALU	(22,16,22)	s	-	-	-	-	-	5	17	34
		k'	-	-	-	-	-	20	16	12

Table 2 presents the required number of segmentation cells for several different cone sizes for various circuits. The experiments were performed on six circuits, viz. combinational benchmark circuits *c*432, *c*499 and *c*880, a bit-sliced 16-bit adder, a 8-bit multiplier and an ALU. The value for the maximum output cone size (r) varies between 40 and 12. The required number of segmentation cells (s) to achieve the desired cone sizes are listed in the table. After partitioning with the segmentation cells, the resulting maximum dependency (k') for the partitioned circuits are also listed. The difference between the desired cone size (r) and the maximum dependency (k') obtained after partitioning with (s) cells is due to the nature of the circuit. For example, circuit *c*499 requires six segmentation cells for reducing the maximum dependency to 32. However, after placing another cell, the maximum dependency of the circuit reduces to 22. This drastic reduction is due to the presence of appropriately placed articulation node in the circuit.

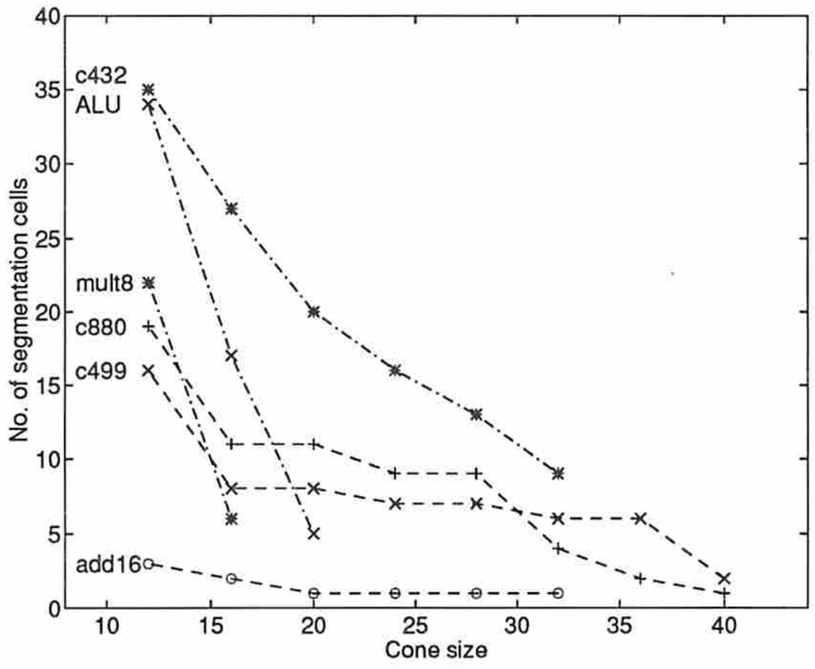

Figure 6: Segmentation cells required for various cone sizes

The data in Table 2 is graphically depicted in Figure 6. The graph illustrates the variations in the required number of segmentation cells as per the variations in the desired cone size. Circuits *c*432, *mult*8 and ALU have steep curves while the circuits *c*499, *c*880 and *add*16 have curves with both steep and flat portions.

The steep behavior is due to the drastic increase in the required number of segmentation cells as the desired cone size is reduced. We believe this behavior can be attributed to the following reasons. First, the scarcity of articulation nodes and/or poor articulation values of nodes in a circuit. For this case, segmentation cells are placed on many reconvergent fanout paths in the circuit in order to reduce the dependencies on the exceeding output cones. Secondly, the exceeding output cones of the circuit may be disjoint and hence may require separate cells to reduce its dependency.

The flat behavior illustrates that no additional segmentation cells are required for the reduction in desired cone size. Circuits with flat behavior have well placed articulation nodes with good articulation values. For example, *add*16 is a bit-sliced circuit, and has articulation nodes connecting the adjacent bit-slices that naturally form candidates for the segmentation cells. Hence the circuit exhibits an almost flat curve.

Note that the curves are extrapolated as straight lines between the actual data points. This extrapolation need not necessarily reflect the true values between the data points. The circuits should ideally give rise to monotonically decreasing curves if experimented with all possible values of desired cone size (r). However, anomalies of the greedy heuristic procedure may result in deviations from the expected behavior.

5 Conclusions

We have presented several strategies to partition combinational circuits for pseudo-exhaustive testing. Optimal solutions to small circuits can be obtained by solving the ILP formulation. Good sub-optimal solutions can be achieved for large circuits using our heuristic procedures. The efficiency and quality of our heuristic approach is demonstrated on the benchmark circuits. With minor modifications, the partitioning strategy can also be applied for sequential circuits at the register transfer level. Our current work involves designing efficient test pattern generators for applying pseudo-exhaustive tests to the partitioned circuits.

References

[1] S. Hellebrand and H-J. Wunderlich. Tools and Devices Supporting the Pseudo-Exhaustive Test. In *Proc. 1st European Design Automation Conf.*, pages 13–17, March 1990.

[2] E. J. McCluskey and S. Bozorgui-Nesbat. Design for Autonomous Test. *IEEE Trans. on Computers*, C-30(11):866–875, November 1981.

[3] M. W. Roberts and P. K. Lala. An Algorithm for the Partitioning of Logic Circuits. *IEE Proc.*, 131(4), July 1984.

[4] S. N. Bhatt, F. R. K. Chung, and A. L. Rosenburg. Partitioning Circuits for Improved Testability. In *Proc. 4th MIT Conf. on Advanced Research in VLSI*, pages 91–106, April 1986.

[5] 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.

[6] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice Hall, Inc., 1974.

[7] 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.

[8] Rajiv Gupta, W. H. Cheng, Rajesh Gupta, I. Hardonag, and M. A. Breuer. An Object-Oriented VLSI CAD Framework: A Case-Study in Rapid Prototyping. *IEEE Computer*, 22(5):28–37, May 1989.

An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing

Rajagopalan Srinivasan, Sandeep K. Gupta
and Melvin A. Breuer

CENG Technical Report 93-08

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4469