POLYNOMIAL TIME SOLVABLE FAULT DETECTION PROBLEMS

Srimat T. Chakradhar
Department of Computer Science & CAIP Research Center, Rutgers University, Piscataway, NJ 08855-1390
Vishwani D. Agrawal
AT&T Bell Laboratories, Murray Hill, NJ 07974
Michael L. Bushnell
CAIP Research Center, Rutgers University, Piscataway, NJ 08855-1390

ABSTRACT – The problem of fault detection in general combinational circuits is NP-complete. Very little work has been done on identifying circuits for which the fault detection problem is polynomial time solvable. The only significant reported result is due to Fujiwara who presented a polynomial time algorithm for detecting any single stuck fault in K−bounded circuits. Such circuits may only contain logic blocks with no more than K input lines and the blocks are so connected that there is no reconvergent fanout among them. We introduce a new class of combinational circuits called the (k, K)−circuits and present a polynomial time algorithm to detect any single or multiple stuck fault in such circuits. Any circuit can be represented as an undirected graph G, with a vertex for each gate in the circuit and an edge between a pair of vertices whenever the corresponding gates have a connection. G for a (k, K)−circuit is a subgraph of a k−tree, which, by definition, cannot have a clique of size greater than k + 1. Basically, this is a restriction on gate interconnections rather than on the function of gates comprising the circuit. The (k, K)−circuits are a generalization of Fujiwara's K−bounded circuits. We formulate the fault detection problem as an energy minimization problem using the bidirectional neural net model we proposed earlier. A minimizing point of the energy function corresponds to a test. In this paper, we present a polynomial time algorithm to solve the single and multiple fault detection problem for the (k, K)−circuits by recursively eliminating variables in the energy function.

1. INTRODUCTION

The problem of detecting a fault in a general combinational circuit is NP-complete [8] and it is unlikely that a polynomial time algorithm exists for solving it. Consequently, it is of interest to identify circuits for which a polynomial time fault detection algorithm exists. The number of *primary inputs* and the number of signals in the digital circuit are generally considered as the the *input size* for the fault detection problem. Very little work has been done on identifying such circuits. Such work is important for two reasons. First, solving special instances usually provides better insights into the solution methods for the general problem. Second, combinational circuits can be designed as special circuits that are easily testable and easy for redundancy identification.

The previous reported result in this area is due to Fujiwara [6] who defined $K-$*bounded* circuits. He assumed that logic blocks with no more than K input lines are connected so that *no* reconvergent fanout exists among these blocks. Essentially, the undirected graph G, with a vertex for each block and an edge between a pair of vertices whenever the corresponding blocks are connected, is a tree. He showed that for such circuits, the single stuck-at fault detection problem is polynomial time solvable.

In this paper, we define a much larger class of circuits, we call the $(k, K)-$*circuits*, and present a polynomial time algorithm to detect any single or multiple stuck-at fault. The logic blocks in $(k, K)-$circuits are still $K-$bounded, i.e., they have no more than K input lines. However, we permit certain cycles in the associated graph G. Specifically, G is a *partial* $k-$*tree* [9]. Any subgraph of a $k-$*tree* (a graph with no *cliques* [2] of size greater than $k+1$) is a partial $k-$tree. Thus, the main difference between a $K-$bounded circuit and a $(k, K)-$circuit is that the latter can have certain reconvergent fanouts among the blocks while the former has none.

Any combinational circuit can be considered as an interconnection of several sub-circuits called *blocks*. A block can have several inputs and outputs. For example, an inverter is a block with one input and one output and a two-input AND gate is a block with two inputs and one output. In general, a block may have several gates. The *union* ($\cup$) of two blocks is defined as the set of gates in both blocks and an *intersection* ($\cap$) is the set of gates common to both blocks. We define an arbitrary combinational circuit C as an interconnection of blocks $C_1 ... C_t$ such that $C = C_1 \cup C_2 \cup ... \cup C_t$ and $C_i \cap C_j = 0$ where $i \neq j$, $1 \leq i \leq t$ and $1 \leq j \leq t$. We will refer to the set $\pi_1 = \{C_1, C_2, ..., C_t\}$ as a *partition* of C (see Table 1). A circuit can be partitioned in several ways. With a partition π_1, we associate an undirected graph $G_{\pi_1}(V, \mathcal{E})$ having a vertex set V and an edge set $\mathcal{E}$. The graph has a vertex for every block C_i in the partition and there is an edge from vertex i to vertex j if the corresponding

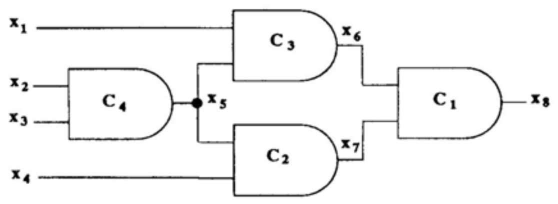

Figure 1: A (2, 2)-circuit.

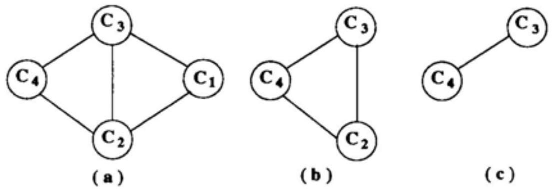

Figure 2: Graphs of circuit in Figure 1. (a) G_{π_1}, (b) G_{π_2}, and (c) G_{π_3}.

blocks have at least one signal in common. For example, the circuit in Figure 1 can be partitioned into four blocks C_1, C_2, C_3 and C_4. Each block has one gate. Furthermore, $\{x_6, x_7, x_8\}$ is the set of signals in block C_1, $\{x_1, x_5, x_6\}$ is the set of signals in C_3 and so on. The corresponding graph G_{π_1} is shown in Figure 2(a). Notice that there is an edge between vertex C_3 and C_2 since they share the signal x_5.

1.1 FUJIWARA'S RESULT

A combinational circuit C is $K-$bounded [6] if it can be partitioned into blocks such that:

1. C_i $(1 \leq i \leq t)$ has at most K inputs and
2. G_{π_1} has no cycles.

For example, the circuit in Figure 3 is 2−bounded since it can be partitioned into three blocks c, g and h, each with at most two inputs. Furthermore, it is easy to see that the corresponding graph does not have a cycle. Also, the circuit in Figure 1 is *not* 2-bounded since the corresponding graph (shown in Figure 2(a)) has a cycle consisting of vertices C_2, C_3 and C_4.

The single stuck-at fault detection problem in $K-$bounded circuits, for a fixed K, is solvable in polynomial time $O(g)$, where g is the number of signals in the circuit [6].

1.2 CONTRIBUTION OF THE PRESENT WORK

In this paper, we relax the restriction on the graph G_{π_1} by allowing certain types of cycles. We define a new class of circuits called the $(k, K)-$*circuits* which can be partitioned into blocks such that

1. C_i $(1 \leq i \leq t)$ has at most K inputs and
2. G_{π_1} is a partial $k-$tree.

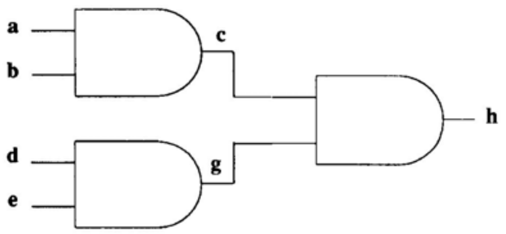

Figure 3: Example logic circuit.

A partial $k-$tree graph is a subgraph of a $k-$*tree* [9]. A $k-$tree is a graph that can be reduced to the $k-$complete graph (i.e., a fully connected graph on k vertices) by a sequence of removals of degree k vertices with completely connected neighbors [9]. A constructive definition of a $k-$tree is given in Section 4. A graph G_{π_1} with no cycles is a special case of a partial $k-$tree (partial 1−tree) and thus, all $K-$bounded circuits are $(1, K)-$circuits.

Note that in $K-$bounded circuits, the parameter K imposes a limit on the number of inputs to a block and these blocks are assumed to be connected such that the associated graph has no cycles. In $(k, K)-$circuits, all blocks still have at most K inputs but the parameter k gives greater freedom in interconnecting the blocks. Observe that we are imposing a restriction on gate interconnections rather than on the function of gates in the circuit. Fixed values of k and K specify a class of circuits. By varying the two parameters, a whole family of circuits can be defined.

We show that the single and multiple stuck-at fault detection problem is solvable in polynomial time for $(k, K)-$circuits. Our proposed algorithm for doing this is radically different from the traditional fault detection algorithms [1]. Using the recently proposed neural net model [4, 5] of logic circuits, we formulate fault detection as an energy minimization problem and show that a minimizing point of the energy function can be determined in time complexity that is a polynomial function of the size of the circuit.

Section 2 reviews the neural net modeling of logic circuits [4]. Section 3 presents the formulation of the fault detection problem as an energy minimization problem. Section 4 reviews partial $k-$tree graphs and discusses notation and terminology used in this paper. In Section 5, we present a polynomial time algorithm for detecting any single or multiple fault in a $(k, K)-$circuit.

2. LOGIC CIRCUIT MODELING

Our model represents the function of a digital circuit as an interconnection of computing elements called neurons.

2.1 DEFINITION OF A NEURON

A *binary neuron* is a computing element that can assume one of two possible states: 0 or 1. A neural net is an interconnection of neurons. We use Hopfield neural networks [7] in which the connections are bidirectional. Thus, neuron i may be connected to another neuron j by a link which is characterized by a weight T_{ij}, where $T_{ij} = T_{ji}$. Each neuron in the network has a *threshold*. A neuron computes its state as follows: (1) it computes a weighted sum of the states of its neighboring neurons and (2) if the result exceeds its own threshold then it sets itself to the 1 state, otherwise it enters the 0 state.

2.2 MODEL FOR A BOOLEAN GATE

Figure 4 shows an AND gate and its neural net model. Each signal is represented by a neuron having a threshold.

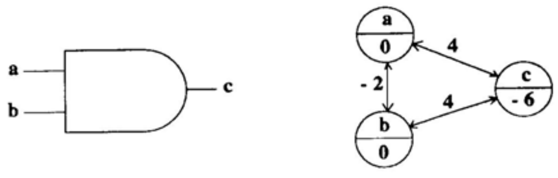

Figure 4: AND gate and its neural net model.

The neurons for signals a and b have 0 thresholds and the neuron for signal c has a threshold of -6. All links are bidirectional and their labels show their weights. The energy function of this three-neuron net is written as [4]:

$$E_{AND}(a, b, c) = -4c(a + b) + 2ab + 6c \qquad (1)$$

Variables a, b, and c can assume only binary values. All operations are arithmetic and not Boolean. It is easily verified that only those values of a, b, and c that conform to the truth table of the AND gate will satisfy $E_{AND} = 0$. Similar models and the corresponding energy functions are derived for all other logic gates [4, 5].

2.3 CIRCUIT MODELING WITH NEURONS

Consider the three gate circuit in Figure 3. Each AND gate has an energy function given by Eqn. (1). The energy function for the entire circuit is obtained by summing the individual gate energy functions. Thus,

$$E_{CKT} = E_{AND}(a, b, c) + E_{AND}(d, e, g) + E_{AND}(c, g, h) \qquad (2)$$

This simple procedure can model circuits with any connectivity and any types of gates. We have shown the existence of neural net models for all logic circuits [4]. Since the models are defined only for two-input gates, gates with larger fan-in are represented as combinations of two-input gates. Any solution of $E_{CKT} = 0$ is a consistent set of signals for the entire circuit. For example, logic simulation will require a solution of this equation when the primary input signals assume given values.

3. TEST GENERATION VIA ENERGY MINIMIZATION

A formulation of the test generation problem as an energy minimization problem requires that for any given fault, we create a copy of the sub-circuit affected by the fault [4]. The corresponding outputs of the *fault-free* circuit and the *faulty* circuit are passed through exclusive-OR gates. The outputs of these exclusive-OR gates are fed to an OR gate whose output is constrained to be 1. In addition, for a s-a-0(1) fault, the signals at the fault site in the fault-free and faulty circuits are constrained to be 1(0) and 0(1), respectively. Any set of consistent signal values now corresponds to a test.

This formulation captures the *necessary* and *sufficient* conditions that any set of signal values must satisfy to be a test: First, the set of values must be consistent with all gate functions in the circuit. Second, the two signals in the fault-free and faulty circuits at the fault site must assume opposite values (e.g., 0 and 1 respectively, for a s-a-1 fault). Third, for the same primary input vector, the fault-free and faulty circuits should produce different output values.

4. NOTATION AND TERMINOLOGY

Consider a combinational circuit C consisting of an interconnection of inverters and 2-input Boolean gates. Let the total number of signals (primary inputs and gate outputs) be g. Let $\pi_1 = \{C_1, C_2, ..., C_t\}$ be the partition associated with the circuit C. We will refer to C_i as the ith block of C. A block consists of one or more gates. An input or output signal of a block is called a *port signal* and a signal that is not an input or output signal of a block is called an *internal signal*. Let g_i be the number of signals in the ith block. For example, in Figure 1, x_1, x_5 and x_6 are the port signals of block C_3. None of the blocks have internal signals.

Consider a graph $G(V, \mathcal{E})$, where V is the vertex set and $\mathcal{E}$ the edge set of G. $G \setminus v$ will denote the subgraph induced by the set $V \setminus v$ (i.e., all vertices except v) where v is a vertex in the graph. A fully connected graph on k vertices is called a *$k-$complete* graph [2]. A $k-$tree [9] is a graph that can be reduced to the $k-$complete graph by a sequence of removal of degree k vertices with completely connected neighbors. This vertex removal sequence is called the *$k-$perfect elimination scheme* of the $k-$tree. For example, the graph in Figure 2(a) is a 2-tree because it can be reduced to a 2-complete graph as follows (see Figures 2(b) and 2(c)): Vertex C_1 is a degree 2 vertex with completely connected neighborhood comprising of vertices C_2 and C_3. If we remove C_1 then C_2 becomes a degree 2 vertex with completely connected neighborhood comprising of vertices C_3 and C_4. Removal of vertex C_2 leaves us with a 2-complete graph consisting of the single edge (C_3, C_4). The sequence of removing C_1 and C_2 is a perfect elimination scheme for this graph. In general, a graph can have several perfect elimination schemes. For example, C_4, C_1 is another perfect elimination scheme of the graph in Figure 2(a).

The class of $k-$trees is defined recursively as follows [9]:

1. The complete graph with k vertices is a $k-$tree.

Table 1: List of symbols.

g	number of signals in circuit C.
t	number of blocks in circuit C.
C_i	ith block in circuit C.
g_i	number of signals in block C_i.
π_i	the partition of circuit C minus the first $i-1$ blocks, i.e., the set $\{C_i, C_{i+1}, ..., C_t\}$.
G_{π_i}	undirected graph corresponding to partition π_i.
N_i	set of neighbors of block C_i in graph G_{π_i}.
J_i	set of variables of all blocks in N_i except the variables in blocks $C_1, C_2, ..., C_{i-1}$.

2. A k−tree with $n+1$ vertices ($n \geq k$) can be constructed from a k−tree with n vertices by adding a vertex adjacent to all vertices of one of its k−vertex complete subgraphs.

For example, the graph in Figure 2(a) can be constructed from the 2-complete graph consisting of the edge (C_3, C_4) as follows. Add vertex C_2 adjacent to all vertices in the 2-complete graph of vertices C_3 and C_4. This results in the edges (C_2, C_4) and (C_2, C_3). Similarly, add vertex C_1 adjacent to all vertices in the 2-complete subgraph consisting of the edge (C_2, C_3). This results in the edges (C_2, C_1) and (C_3, C_1).

A partial k−tree is a subgraph of a *k−tree.* For example, consider the 2-tree graph in Figure 2(a). The subgraph with vertices C_1, C_2 and C_3 is a partial 2-tree. So is the subgraph with vertices C_1, C_3 and C_4. Obviously, since the entire graph can be treated as a subgraph, the entire graph is a partial 2-tree. The perfect elimination scheme for the partial k−tree is the same as the elimination sequence for the corresponding k−tree. The problem of finding the smallest number k such that a given graph is a partial k-tree is NP-complete [3]. Table 1 lists the symbols used in the sequel.

5. A POLYNOMIAL TIME TEST GENERATION ALGORITHM

We will first consider the problem of detecting a primary output fault in a (k, K)−circuit and then show how an arbitrary single or multiple fault in a (k, K)−circuit can be detected in polynomial time, $O(g^2)$.

5.1 PRIMARY OUTPUT FAULT

Consider a single-output circuit C with a s-a-0(1) fault on the primary output. The test must simply control the output to 1(0). It is, therefore, unnecessary to create a faulty circuit copy. We constrain the primary output signal to assume the value 1(0). Let E_i be the energy function associated with the ith block. If the block has more than one gate, then E_i is the sum of the individual gate energy functions. The energy function for the entire circuit is $E_{CKT} = E_1 + E_2 + ... + E_t$. We can simplify E_{CKT} by substituting the value of the primary output signal. Now, any solution of $E_{CKT} = 0$ is a consistent set of signals for the entire circuit and, hence, corresponds to a test for the fault. If no solution is possible, then the fault is redundant.

To show that a minimizing point of E_{CKT}, and hence, a test for the fault, can be obtained in polynomial time, we will need the following lemmas.

Lemma 1: *Let C be a (k, K)−circuit with a partition $\pi_1 = \{C_1, C_2, ..., C_t\}$ and let $(C_1, C_2, ..., C_{t-k})$ be the k−perfect elimination scheme of the graph G_{π_1}. Let $E_{1,t} = E_{CKT}$ be the energy function for C. The minimization $E_{1,t}$ can be reduced in $O(g)$ time to the minimization of a function $E_{2,t}$ that depends on variables in blocks $C_2, C_3, ..., C_t$ but not on the port variables of block C_1. Also, the function $E_{2,t}$ has at most a constant number (2^{kK}) of terms more than the function $E_{1,t}$.*

Proof: In order to construct the function $E_{2,t}$ from $E_{1,t}$, we explicitly write the terms of $E_{1,t}$ corresponding to the variables in C_i ($1 \leq i \leq t-k$) that are excluded from $C_1, C_2, ..., C_{i-1}$. We also construct a list of terms that involve variables in blocks C_i, $t-k+1 \leq i \leq t$, but not in the first $t-k$ blocks. The function $E_{1,t}$ has $O(g)$ terms since each gate contributes at most six terms [5]. Our representation can be obtained in $O(g)$ time by scanning through the terms of $E_{1,t}$.

We can rewrite the function $E_{1,t}$ as:

$$E_{1,t} = f_1 + h_1$$

where the function h_1 contains all terms that do not involve variables in block C_1 and f_1 contains only terms involving variables in block C_1. Let $N_1 = \{i : C_i$ is connected to $C_1\}$ be the set of neighbors of block C_1 in G_{π_1}. Since G_{π_1} is a partial k−tree, block C_1 is connected to at most k other blocks. Therefore, f_1 has at most $\sum_{i \in N_1} g_i$ variables other than the variables in block C_1. Let J_1 be the set of these $\sum_{i \in N_1} g_i$ variables. Note that in any consistent labeling of the circuit C, all blocks C_i ($1 \leq i \leq t$) are also consistently labeled. Although a block has g_i variables, there are at most only a constant number (2^K) of consistent labelings of the signals in block C_i, because the block has at most K inputs. Similarly, there can be at most 2^{kK} consistent labelings of the blocks $\{C_i : i \in N_1\}$. Therefore, the $\sum_{i \in N_1} g_i$ variables in set J_1 can assume at most a constant number (2^{kK}) of combinations of Boolean values. For a given vector α of values for the variables in the set J_1, we can evaluate f_1 for all 2^K combinations of values of the variables in block C_1. Let $f_1(\alpha)$ denote the minimum value the function f_1 attains for the vector α.

Clearly, there exists a minimizing point of function $E_{1,t}$, say (β^*, α^*), where β^* is a vector of values for the variables in block C_1 and α^* is a vector of values for variables in the set J_1, such that the minimum value of f_1 for the given vector α^* is attained at β^*. Also, β^* is a *non-zero vector* (i.e., a vector with at least one non-zero component) *iff* $f_1(\alpha^*) < 0$. This leads us to define a function ψ_1 that depends only on the $\sum_{i \in N_1} g_i$ variables in set J_1, such that

$$\psi_1 = \sum_{\alpha \in M_1} \min(0, f_1(\alpha)) \times T_\alpha \qquad (3)$$

where M_1 is the set of a constant number (2^{kK}) of consistent labelings (vectors) of the blocks $\{C_i : i \in N_1\}$ and T_α is the product of literals corresponding to vector α (i.e., $T_\alpha = \prod_{l \in J_1} l_\alpha$ where $l_\alpha = l$ if variable l is 1 in vector α and $l_\alpha = \bar{l}$ if $l = 0$). We evaluate f_1 at all combinations of values of variables in J_1 and $\min(0, f_1(\alpha))$ is the minimum value the function f_1 attains, given a particular vector α of values for the variables in J_1. Therefore, functions f_1 and ψ_1 have the same minimum value.

Let

$$E_{2,t} = \psi_1 + h_1 \qquad (4)$$

We have thus reduced the problem of minimization of the original function $E_{1,t}$ that depends on all the blocks C_i $(1 \le i \le t)$ to the minimization of $E_{2,t}$, that only depends on blocks C_i $(2 \le i \le t)$. Since, block C_1 is eliminated, the partition corresponding to $E_{2,t}$ is $\pi_2 = \{C_2, C_3, ..., C_t\}$. A minimizing point $(\beta^*\alpha^*)$ of $E_{1,t}$ can then easily be traced back from any minimizing point α^* of $E_{2,t}$ as explained later. Also, since ψ_1 can have at most a constant number (2^{kK}) of terms, $E_{2,t}$ can have at most a constant number of terms more than $E_{1,t}$ and the representation of $E_{2,t}$ can be obtained in $O(g)$ time. ∎

Example: Consider the s-a-1 fault on signal x_8 in the circuit of Figure 1. We construct the energy function for the circuit and find a minimizing point by recursively eliminating the variables of blocks in the order C_1, C_2, C_3, C_4. Here, we illustrate the elimination of variables of block C_1 from the energy function of the circuit. The elimination of other blocks is discussed later. The energy function for the circuit is:

$$\begin{aligned} E_{CKT} &= E_1 + E_2 + E_3 + E_4 \\ &= -4x_8(x_6 + x_7) + 2x_6x_7 + 6x_8 \\ &\quad -4x_7(x_4 + x_5) + 2x_4x_5 + 6x_7 \\ &\quad -4x_6(x_1 + x_5) + 2x_1x_5 + 6x_6 \\ &\quad -4x_5(x_2 + x_3) + 2x_2x_3 + 6x_5 \end{aligned}$$

Since signal x_8 is constrained to assume the value 0, we can simplify E_{CKT} by substituting $x_8 = 0$. Let $E_{1,4}$ be the simplified E_{CKT}. The first row in Table 2 gives the terms of $E_{1,4}$ that involve variables in block C_1 (i.e., x_6, x_7 and x_8). The second row gives the terms involving the variables of block C_2 that are not in block C_1 (i.e., x_4 and x_5). The last row gives the terms that have only

Table 2: Terms of $E_{1,4}$.

Block	Terms
C_1	$2x_6x_7 + x_6(6 - 4x_1 - 4x_5) + x_7(6 - 4x_4 - 4x_5)$
C_2	$2x_4x_5 + x_5(6 + 2x_1 - 4x_2 - 4x_3)$
C_3, C_4	$2x_2x_3$

Table 3: Possible values of f_1.

x_1	x_4	x_5	f_1	$\min f_1$
0	0	0	$2x_6x_7 + 6(x_6 + x_7)$	0
0	0	1	$2x_6x_7 + 2(x_6 + x_7)$	0
0	1	0	$2x_6x_7 + 6x_6 + 2x_7$	0
0	1	1	$2x_6x_7 + 2x_6 - 2x_7$	-2
1	0	0	$2x_6x_7 + 2x_6 + 6x_7$	0
1	0	1	$2x_6x_7 - 2x_6 + 2x_7$	-2
1	1	0	$2x_6x_7 + 2x_6 + 2x_7$	0
1	1	1	$2x_6x_7 - 2x_6 - 2x_7$	-2

those variables of the last two blocks C_3 and C_4 that are not in blocks C_1 and C_2. The first row in Table 2 is the function f_1. The terms in second and third row belong to the function h_1. $N_1 = \{C_2, C_3\}$ is the set of neighbors of block C_1. The set of variables in f_1 that are not in block C_1 is $J_1 = \{x_1, x_4, x_5\}$. Table 3 shows the minimum value of f_1 for each combination of values of the variables in set J_1. For example, when $x_1 = 0$, $x_4 = 0$ and $x_5 = 0$, $f_1 = 2x_6x_7 + 6(x_6 + x_7)$ and it assumes a minimum value 0 when $x_6 = x_7 = 0$. Only the fourth, sixth and eighth rows of $\min f_1$ contribute to ψ_1. We multiply the $\min f_1$ value by the corresponding x_1, x_4 and x_5 literals. From Eqn. 3,

$$\begin{aligned} \psi_1 &= -2(\overline{x_1}x_4x_5 + x_1\overline{x_4}x_5 + x_1x_4x_5) \\ &= -2(\overline{x_1}x_4x_5 + x_1x_5) \end{aligned} \qquad (5)$$

and from Eqn. 4,

$$E_{2,4} = \psi_1 + h_1 \qquad (6)$$

The terms of $E_{2,4}$ are obtained from $E_{1,4}$ (Table 2) as follows. Remove the row for block C_1. Since the two terms in ψ_1 (Eqn. 5) involve variables from block C_2, add these two terms to row C_2. The complete $E_{2,4}$ is shown in Table 4. This eliminates block C_1. Thus, minimization of $E_{1,4}$ is reduced to minimization of $E_{2,4}$. Removal of vertex C_1 from the partition π_1 gives the new partition $\pi_2 = \{C_2, C_3, C_4\}$ and the corresponding graph G_{π_2} is shown in Figure 2(b).

Lemma 2: *If graph G_{π_1} is a partial $k-$tree with elimination sequence $(C_1, C_2, C_3, ..., C_{t-k})$ then G_{π_2} is also a partial $k-$tree with elimination sequence $(C_2, C_3, ..., C_{t-k})$.*

Proof: Let H denote a $k-$tree containing G_{π_1}, the graph corresponding to function $E_{1,t}$. Also, let $(C_1, C_2, C_3, ..., C_{t-k})$ be a $k-$perfect elimination scheme

Table 4: Terms of $E_{2,4}$.

Block	Terms
C_2	$2x_4x_5(1-\overline{x_1}) + x_5(6-4x_2-4x_3)$
C_3, C_4	$2x_2x_3$

of H and $N_1 = \{i : C_i$ is connected to $C_1\}$ be the set of neighbors of block C_1 in G_{π_1}. Notice that N_1 induces a complete subgraph in H. Moreover, G_{π_2} is identical to $G_{\pi_1} \setminus C_1$, except possibly for some edges between vertices of N_1 (since ψ_1 depends only on the variables in N_1). Therefore, G_{π_2} is also a subgraph of $H \setminus C_1$, and the claim follows. ∎

Lemma 3: *A minimizing point of the function $E_{1,t}$ can be obtained in $O(g^2)$ time.*

Proof: Using Lemma 1, we can eliminate block C_1 to generate the new function $E_{2,t}$ in $O(g)$ time. From Lemma 2, the elimination sequence for the corresponding graph G_{π_2} is $(C_2, C_3, ..., C_t)$. Continuing the elimination process in Lemma 1 for blocks $C_2, C_3, ..., C_{t-k}$, successively, we produce two sequences of functions $E_{2,t}, E_{3,t}, ..., E_{t-k+1,t}$ and $\psi_1, \psi_2, ..., \psi_{t-k}$ where $E_{i,t}$, $(1 \le i \le t-k+1)$ does not depend on any variable in blocks $C_1, C_2, ..., C_i$. From Lemma 1, elimination of a single block involves $O(g)$ work. Since the number of blocks is less than g, the work in eliminating the blocks $\{C_i : 1 \le i \le t-k\}$ is bounded by $O(g^2)$. The function $E_{t-k+1,t}$ depends at most on k blocks and the minimum value and a minimizing point of $E_{t-k+1,t}$ can be obtained by examining at most a constant number (2^{kK}) of combinations of values for the variables in the k blocks that E_{t-k+1} depends on. This can be done in time $O(g)$. Note that the minimum value of the function E_{t-k+1} is also the minimum value of $E_{1,t}$. Given a minimizing point for the function $E_{i+1,t}$, a minimizing point for function $E_{i,t}$ can be obtained in $O(g)$ time using back substitution and, therefore, a minimizing point for $E_{1,t}$ can be obtained in $O(g^2)$ from the minimizing point of $E_{t-k+1,t}$. Hence, a minimizing point of $E_{1,t}$ can be obtained in $O(g^2)$ time. ∎

Continuing the previous example, a minimizing point of $E_{1,4}$ is found by eliminating the remaining blocks. C_2 is eliminated first. The row for C_2 in Table 4 gives f_2. The sum of the terms in the remaining rows is h_2. $N_2 = \{C_3, C_4\}$ is the set of neighbors of block C_2 and $J_2 = \{x_1, x_2, x_3\}$. The minimum value of f_2 for each combination of values of variables in J_2 is given in Table 5. Therefore, from Eqn. 3,

$$\begin{aligned} \psi_2 &= -2x_2x_3(x_1 + \overline{x_1}) \\ &= -2x_2x_3 \qquad (7) \end{aligned}$$

and from Eqn. 4,

$$\begin{aligned} E_{3,4} &= \psi_2 + h_2 \\ &= -2x_2x_3 + 2x_2x_3 \\ &= 0 \end{aligned}$$

Minimization of $E_{2,4}$ is now reduced to the minimization of $E_{3,4}$ which does not depend on the variables x_4 and x_5. In fact, $E_{3,4} = 0$ and hence, it does not depend on any variable. Clearly, the minimum value of $E_{3,4}$ is 0. Values of variables x_1, x_2 and x_3 are normally determined from $E_{3,4}$. However, since $E_{3,4} = 0$, they can assume any values, say, $x_1 = x_2 = x_3 = 1$ and there exists a test for the output s-a-1.

With the elimination phase over, we proceed to find a minimizing point of $E_{2,4}$ from a minimizing point of $E_{3,4}$. From Table 5, $f_2(x_1 = x_3 = x_4 = 1) = 2x_4x_5 - 2x_5$ which assumes a minimum value of -2 when $x_4 = 0$ and $x_5 = 1$. Hence, $x_1 = x_2 = x_3 = x_5 = 1$, $x_4 = 0$ is one minimizing point of $E_{2,4}$. Similarly, we can find a minimizing point of $E_{1,4}$ from that of $E_{2,4}$. From Table 3, $f_1(x_1 = 1, x_4 = 0, x_5 = 1) = 2x_6x_7 - 2x_6 + 2x_7$ which assumes a minimum value of -2 when $x_6 = 1$ and $x_7 = 0$. Therefore, $x_1 = x_2 = x_3 = x_5 = x_6 = 1$, $x_4 = x_7 = 0$ is a minimizing point of $E_{1,4}$ or E_{CKT}. The input $x_1 = x_2 = x_3 = 1$, $x_4 = 0$ is a test for the primary output s-a-1 fault.

Theorem 1: *There exists a polynomial time $(O(g^2))$ algorithm that either detects any primary output fault in a $(k, K)-$circuit or identifies the fault as redundant.*

Proof: We give a constructive proof. Let C be a $(k, K)-$circuit with a partition $\pi_1 = \{C_1, C_2, ..., C_t\}$ and let $(C_1, C_2, ..., C_{t-k})$ be the $k-$perfect elimination scheme of the graph G_{π_1}. The algorithm is as follows:

1. Construct the energy function E_{CKT} corresponding to the stuck-at primary output fault in circuit C. This can be done in $O(g)$ time [4].
2. Find the minimum value of E_{CKT} and the corresponding minimizing point. Using Lemma 3, this can be done in $O(g^2)$ time.
3. Check whether minimum value of $E_{CKT} = 0$. If so, the minimizing point corresponds to a test. Otherwise, identify the fault as redundant.

Table 5: Possible values of f_2.

x_1	x_2	x_3	f_2	$\min f_2$
0	0	0	$6x_5$	0
0	0	1	$2x_5$	0
0	1	0	$2x_5$	0
0	1	1	$-2x_5$	-2
1	0	0	$2x_4x_5 + 6x_5$	0
1	0	1	$2x_4x_5 + 2x_5$	0
1	1	0	$2x_4x_5 + 2x_5$	0
1	1	1	$2x_4x_5 - 2x_5$	-2

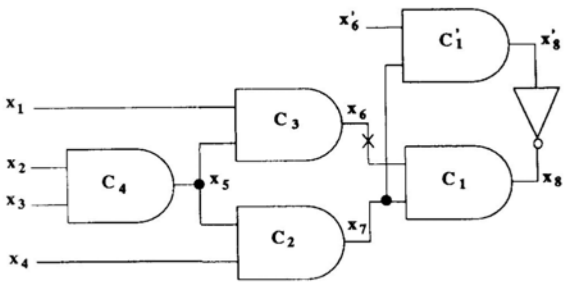

Figure 5: ATPG circuit for x_6 s-a-0.

Clearly, the above algorithm runs in $O(g^2)$ time. ∎

5.2 ARBITRARY SINGLE FAULT

We assume that the (k, K)−circuit C only has one primary output. The case of more than one primary outputs will be discussed later. To detect an arbitrary fault in C, we construct the fault-free and faulty circuit as described in Section 3. Since the circuit has just one output, we can avoid the use of exclusive-OR gates. An inverter between the outputs of the fault-free and faulty circuit will ensure that the two circuits produce different outputs [4]. This inverter is treated as a part of the block that contains the primary output signal. This circuit, as shown in Figure 5 is called the *ATPG circuit.* Note that the inverter is used only for the neural net model which contains bidirectional links.

Lemma 4: *The ATPG circuit is a $(k, 2K)$-circuit.*

Proof: We will show that the ATPG circuit is a $(k, 2K)$-circuit by constructing a partition π such that G_π is a partial k−tree graph. Let $\pi_1 = \{C_1, C_2, ..., C_t\}$ be the partition of the fault-free (k, K)−circuit C. We partition the ATPG circuit as follows: Observe that for every block C_i, in the fault-free circuit, that lies on a path from the fault site to the primary output, there is a corresponding block C_i' in the faulty circuit. We merge these two blocks into one block M_i. Therefore, $M_i = C_i \cup C_i'$. Since C_i and C_i' each have at most K inputs, the new block M_i will have at most $2K$ inputs. Also, if a block C_j is not on a path from the fault site to a primary output, $M_j = C_j$. Therefore, $\pi = \{M_1, M_2, ..., M_t\}$ is a partition of the ATPG circuit and G_π is clearly a partial k−tree graph since it is isomorphic to the partial k−tree graph G_{π_1}. ∎

Example: The ATPG circuit for a s-a-0 fault on signal x_6 in Figure 1 is shown in Figure 5. Only block C_1 is on a path from the fault site to the primary output. The corresponding block in the faulty circuit is C_1'. The graph $G_{\pi'}$ corresponding to the partition $\pi' = \{C_1, C_2, C_3, C_4, C_1'\}$ is shown in Figure 6(a). Merging blocks C_1 and C_1' causes the edges (C_1, C_1') and (C_2, C_1') in graph $G_{\pi'}$ to disappear. Also, $M_1 = C_1 \cup C_1'$. Since none of the other blocks in

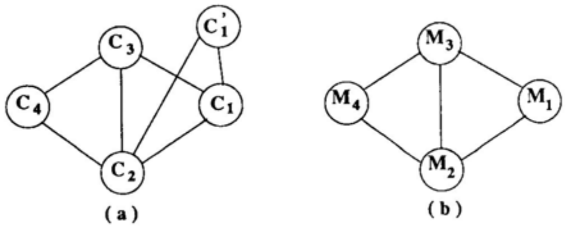

Figure 6: Graphs of the single fault ATPG circuit. (a) $G_{\pi'}$ and (b) G_π.

the fault-free circuit are on a potential fault propagation path, $M_2 = C_2$, $M_3 = C_3$ and $M_4 = C_4$. The graph G_π corresponding to the partition $\pi = \{M_1, M_2, M_3, M_4\}$ is shown in Figure 6(b). Clearly, G_π is isomorphic to the partial k−tree graph G_{π_1} in Figure 2(a).

Theorem 2: *There exists a polynomial time ($O(g^2)$) algorithm that either detects any arbitrary given fault in a (k, K)−circuit or identifies the fault as redundant.*

Proof: Let signal x_i be s-a-0(1). The following algorithm either detects the fault or identifies it as redundant:

1. Construct the energy function E_{ATPG} corresponding to the ATPG circuit for the fault. Simplify E_{ATPG} by substituting $x_i = 1(0)$ and $x_i' = 0(1)$ where x_i and x_i' are the signals in the fault-free and faulty circuits, respectively, corresponding to the fault site. Also, from Lemma 4, the ATPG circuit is a partial k−tree with $(M_1, M_2, ..., M_{t-k})$ as a perfect elimination scheme.

2. Find a minimizing point of E_{ATPG}. From Lemma 3, such a point can be obtained in $O(g^2)$ time.

3. Check whether at the minimizing point $E_{ATPG} = 0$. If so, the minimizing point corresponds to a test. Otherwise, identify the fault as redundant.

Clearly, the above algorithm runs in $O(g^2)$ time. ∎

Multi-output circuits: For (k, K)−circuits with more than one primary output, we repeat the above procedure for each primary output that is reachable from the fault site. Since the number of primary outputs of the circuit is bounded by g (the number of signals in the circuit), the procedure may have to be repeated only a polynomial number of times. An alternate method would be to construct the ATPG circuit for the multi-output circuit [4], find an elimination scheme, and use the procedure just once to generate a test. Due to space limitations, we will not discuss this further.

5.3 MULTIPLE FAULTS

We again assume that the (k, K)−circuit C only has one primary output. The case of more than one primary outputs will be discussed later. To detect a mul-

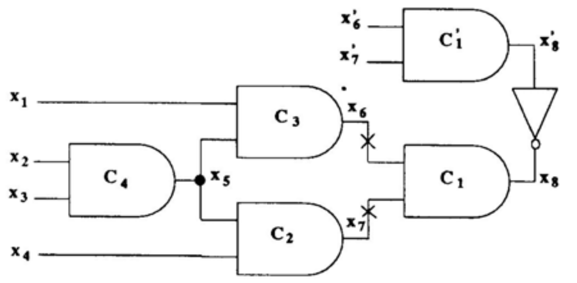

Figure 7: ATPG circuit for the multiple fault x_6 s-a-0, x_7 s-a-0.

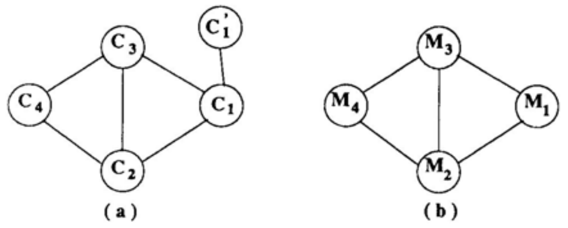

Figure 8: Graphs of the multiple fault ATPG circuit. (a) $G_{\pi'}$ and (b) G_π.

tiple fault in C, we construct the fault-free and faulty circuit in a similar fashion as described in Sections 3 and 5.2 except that we duplicate all blocks that are on the path from any of the multiple fault sites. For example, the ATPG circuit for a multiple fault on signals x_6 and x_7 is shown in Figure 7. Block C_1 is duplicated and $\pi' = \{C_1, C_2, C_3, C_4, C'_1\}$ is a partition of the ATPG circuit. The corresponding graph $G_{\pi'}$ is shown in Figure 8(a). Merging the blocks on the fault propagation paths gives the partition $\pi = \{M_1, M_2, M_3, M_4\}$. The corresponding graph is shown in Figure 8(b). From Lemma 4, it follows that the ATPG circuit is a $(k, 2K)$-circuit.

Theorem 3: *There exists a polynomial time ($O(g^2)$) algorithm that either detects any multiple fault in a (k,K)−circuit or identifies the fault as redundant.*

Proof: The foll›wing algorithm either detects a multiple fault or identifies it as redundant:

1. Construct the energy function E_{ATPG} corresponding to the ATPG circuit for the multiple fault. Simplify E_{ATPG} by substituting appropriate values for the variables in the fault-free and faulty circuits corresponding to the multiple fault sites. Also, from Lemma 4, the ATPG circuit is a partial k−tree with $(M_1, M_2, ..., M_{t-k})$ as the perfect elimination scheme.

2. Find a minimizing point of E_{ATPG}. From Lemma 3, such a point can be obtained in $O(g^2)$ time.

3. Check whether the minimum value of $E_{ATPG} = 0$. If so, the minimizing point corresponds to a test. Otherwise, identify the fault as redundant.

Clearly, the above algorithm runs in $O(g^2)$ time. ∎

Multi-output circuits: For (k,K)−circuits with more than one primary output, we repeat the above procedure for each primary output that is reachable from the fault site. Since the number of primary outputs of the circuit is bounded by g (the number of signals in the circuit), the number of times the above procedure may be repeated is bounded by $O(g)$. An alternate procedure will be to construct the ATPG circuit for the multi-output circuit, find an elimination scheme, and use the procedure just once to generate a test.

6. CONCLUSION

We have identified a class of combinational circuits in which the single and multiple fault detection problem can be solved in polynomial time. A novel test generation method is presented. This is a significant result in test generation and in the identification of redundancies. Work is underway to extend the fault detection algorithm for (k,K)−circuits to general combinational circuits. Also, we are investigating the design of combinational functions as the easier-to-test (k,K)−circuits.

REFERENCES

[1] V. D. Agrawal and S. C. Seth. *Test Generation for VLSI Chips.* IEEE Computer Society Press, Washington, D.C., 1988.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

[3] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of Finding Embeddings in a k-Tree. *SIAM J. on Alg. Disc. Meth.*, 8(2):277–284, April 1987.

[4] S. T. Chakradhar, M. L. Bushnell, and V. D. Agrawal. Automatic Test Pattern Generation Using Neural Networks. In *IEEE Proc. of the Int'l. Conf. on Computer-Aided Design*, pages 416–419, Nov. 1988.

[5] S. T. Chakradhar, M. L. Bushnell, and V. D. Agrawal. Toward Massively Parallel Automatic Test Generation. *IEEE Trans. on Computer-Aided Design*, 1990. *To appear.*

[6] H. Fujiwara. Computational Complexity of Controllability/Observability Problems for Combinational Circuits. In *Proc. of the 18th Int'l. Symp. on Fault Tolerant Computing*, pages 64–69, June 1988.

[7] J. J. Hopfield. Artificial Neural Networks. *IEEE Circuits and Devices Mag.*, 4(5):3–10, Sept. 1988.

[8] O. H. Ibarra and S. K. Sahni. Polynomially Complete Fault Detection Problems. *IEEE Trans. on Computers*, C-24(3):242–249, March 1975.

[9] D. J. Rose. On Simple Characterizations of k-trees. *Discrete Mathematics*, 7(3,4):317–322, Feb. 1974.