Localization of Single Gate Design Errors in Combinational Circuits by Diagnostic Information about Stuck-at Faults

R. Ubar[1], D. Borrione
Université Joseph Fourier/TIMA, 120 Rue de la Piscine, Grenoble, France
raiub@pld.ttu.ee, Dominique.Borrione@imag.fr

Abstract

A new approach to detecting and localizing single gate design errors in combinational circuits is proposed. The method is based on using fault tables for stuck-at fault diagnosis with subsequent translation of the result into the design error area. This allows to exploit standard gate-level ATPGs also for diagnosis of design errors. A powerful hierarchical approach is proposed based on using structurally synthesized BDDs, where combining the error detection and diagnosis phases into a single whole allows to drastically reduce the amount of work for error site localization.

1. Introduction

Digital systems are becoming increasingly complex and therefore, design verification and design error localization are becoming more and more time consuming in case of designs containing hundreds of thousands gates as random logic. Verification and error localization are traditionally handled separately: for verification the methods of simulation and tautology checking can be used, whereas for error localization, after an error is detected, other dedicated methods are introduced [1,2].

While a lot of work has been done in the field of test synthesis and fault diagnosis in relation to fabrication faults, very little has been done in the field of design error diagnosis [1-5]. In [6] a new BDD technique has been proposed, however the explosion of the complexity for some classes of circuits puts practical limitations to the use of BDDs in locating design errors. A brief overview of currently available solutions to the diagnosis problem has been given in [2].

The technique proposed in [1] assumes the existence of a single gate error in the combinational circuit. Simple gate errors are considered, and three error hypotheses have been introduced. The diagnoser works successively under one of these hypotheses. The reasoning will be carried out at the plain gate level. A set of rules has been developed for all procedures with gates concerning the diagnostic reasoning as well as the creation of activated paths through gates.

In the present paper, the same problem is formulated as in [1], i.e. a single design error case in combinational circuits is being attacked. Differently from [1] where the whole analysis is carried out at the plain gate level, in the present paper, a hierarchical approach will be exploited which allows to increase the speed in error detection and localization. Also differently from [1] where only the diagnosis problem is formulated and solved, in the present paper, the error detection and error diagnosis tasks are solved jointly which allows to increase the efficiency of error localization.

The originality of this paper lies in using structurally synthesized BDDs [7] which allowed to develop efficient higher than gate level path activation and fault reasoning procedures for increasing the speed in test generation and fault diagnosis. The method developed in the paper is based on the stuck-at fault model, where all the analysis and reasoning is carried out in terms of stuck-at faults and only in the end, the result of diagnosis will be mapped into the design error area. Such a treatment allows to exploit traditional ATPGs to serve the problem of design error diagnosis.

The paper is organized in the following manner. Section 2 presents the necessary definitions and terminology. The use of stuck-at faults and mapping the diagnosis results into the design error area is explained in Section 3. The error detection technique is given in Section 4, and the error localization ideas are described in Section 5. Some considerations on the efficiency of the approach are brought in Section 6, and Section 7 presents some conclusions.

[1] On the leave from Tallinn Technical University, Raja 15, EE0026 Tallinn, Estonia.

2. Definitions and Terminology

Consider a circuit specification, and its implementation, both at the Boolean level. The specification output is given by a set of variables $W = \{w_1, w_2, \ldots, w_m\}$, and the implementation output is given by a set of variables $Y = \{y_1, y_2, \ldots, y_m\}$, where m is the number of outputs. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of input variables. The implementation is a gate network and Z is the set of internal variables used for the connection of gates. The gates are implementing simple Boolean functions AND, OR, NAND, NOR and NOT. An additional gate type FAN is added (one input, two or more outputs) to model fanout points.

We use two different levels for representing the network: the gate and macro-level representations. Let S be the set of variables in the implementation $S = Y \cup Z \cup X$. Let X^F and Z^F be the subsets of inputs and internal variables that fanout (they are input to a FAN gate). Let Z^{FG} be the subset of internal variables that are output of a FAN gate. Then at the gate level, the network can be described by a set $NG = \{g_k\}$ of gate functions $s_k = g_k(s_k^1, s_k^2, \ldots, s_k^h)$ where $s_k \in Y \cup Z$, and $s_k^j \in Z \cup (X - X^F)$. Let us introduce macro functions for representing tree-like subcircuits of the network. Then, at the macro-level, the network is given by a set $NF = \{f_k\}$ of macro functions $s_k = f_k(s_k^1, s_k^2, \ldots, s_k^p)$ where $s_k \in Y \cup Z^F$, and $s_k^j \in Z^{FG} \cup (X - X^F)$.

Definition 2.1. Test patterns. For a circuit with n inputs, a *test pattern* T is a n-bit vector which may be binary B^n or ternary T^n, where $B = \{0,1\}$ - the Boolean domain, $T = \{0,1,U\}$ - the ternary domain, where U - is a don't care.

Definition 2.2. Stuck-at fault set. Let F be the set of stuck-at-1 faults s/1 and stuck-at-0 faults s/0, where $s \in Z \cup X$. Detection of faults in F is sufficient for stating that the circuit is stuck-at fault free.

Definition 2.3. Detecting stuck-at faults. A test pattern T_i detects a stuck-at-e fault s/e, $e \in \{0,1\}$ at the output y_j, if when applying the test pattern T_i to the implementation and the specification, the result $y_j(T_i) \neq w_j(T_i)$ is observed. Mathematically, a stuck-at-e is detected on s if: $T_i \rightarrow (\partial y/\partial s = 1)\ \&\ (s = \neg e)$, where $s \in Z \cup X$, and $y \in Y$.

Definition 2.4. Stuck-at fault cover. The circuit is tested completely by a test $T = \{T_1, T_2, \ldots T_t\}$ for stuck-at faults, iff T detects all the faults in F. The gate g_k which implements the function $s_k = g_k(s_k^1, s_k^2, \ldots, s_k^h)$ is tested by T for stuck-at faults, iff T detects both stuck-at-1 and stuck-at-0 faults at all the gate inputs s_k^j.

The stuck-at fault model does not have in this paper a physical meaning. In reality, a design error is detected at y_j when under the application of a test T_k, a result $y_j(T_k) \neq w_j(T_k)$ is observed. Using the stuck-at fault model, we only imitate the traditional testing by comparing the behavior of the implementation and the specification as a "golden device". From tests that have shown an error, we produce, as in the case of traditional testing, a diagnosis in terms of stuck-at faults, which are then mapped into design errors. The following design error types are considered throughout the paper in relation to gates $g_k \in NG$.

Definition 2.5. Gate replacement error. It denotes a design error which can be corrected by replacing the gate g_i in NG with another gate g_j, by $g_i \rightarrow g_j$.

Definition 2.6. Extra/missing invertor error. It denotes a design error which can be corrected by removing/inserting an invertor at some input $s \in X$, or at some fanout branch $s \in Z^{FG}$: $s \rightarrow NOT(s)$.

Definition 2.7. Single error hypothesis. Our design error diagnosis methodology is based on a *single error hypothesis* where it is assumed that in the circuit a single error from the following error types can exist: 1) an extra/missing inverter, 2) an arbitrary gate replacement between AND, OR, NAND, NOR gates.

3. Mapping detected stuck-at faults into design errors

Theorem 3.1. To detect a design error in the implementation at an arbitrary gate g_k where $s_k = g_k(s_1, s_2, \ldots, s_h)$, it is sufficient to apply a pair of test patterns which detect the stuck-at faults $s_i/1$ and $s_i/0$ at one of the gate inputs s_i, $i = 1,2, \ldots h$.

Proof. 1. Consider first the detection of AND $\leftrightarrow$ OR errors. A necessary condition is:

$$(s_1 \wedge s_2 \wedge \ldots s_h) \oplus (s_1 \vee s_2 \vee \ldots s_h) = 1. \quad (1)$$

The possible solutions of this equation are

$$s_i \wedge \neg s_j = 1, \text{ where } i,j = 1,2, \ldots h, \text{ and } i \neq j. \quad (2)$$

Thus, if we set at least two inputs of a gate to complementary values, then the errors of types AND $\rightarrow$ OR and OR $\rightarrow$ AND will be detected at the output of the gate.

2. Consider the case of design errors related to the AND gate. Let us choose a test pattern

$$T_{AND,1} = \{s_i = 0, \forall j, j \neq i: s_j = 1\},$$

which is one of solutions (2) of the equation (1), and which detect also the stuck-at fault $s_i/1$ at the AND input. It is easy to see that the pattern $T_{AND,1}$ detects not only the error AND $\rightarrow$ OR, but also the error AND $\rightarrow$ NAND, and the errors of missing/extra invertors at the input s_i.

Consider now the error AND $\rightarrow$ NOR. The necessary condition for detecting the error can be formulated as:

$$(s_1 \wedge s_2 \wedge \ldots s_h) \oplus \neg (s_1 \vee s_2 \vee \ldots s_h) = 1, \quad (3)$$

which has two solutions:

$$(s_1 \wedge s_2 \wedge \ldots s_h) = 1, \quad (4)$$

$$(\neg s_1 \wedge \neg s_2 \wedge \ldots \neg s_h) = 1. \quad (5)$$

The solution (4) gives a test pattern $T_{AND,2} = \{\forall i,\ i = 1,2, \ldots h: s_i = 1\}$,

which detects not only the design error AND $\rightarrow$ NOR, but also stuck-at faults s_i /0 at all of the AND inputs. It is easy to see that the pattern $T_{AND,2}$ detects also all the errors of missing/extra invertor at the other AND gate inputs s_j, $j \neq i$ which were not detected by the pattern $T_{AND,1}$.

Hence, we have shown that the test patterns $T_{AND,1}$ and $T_{AND,2}$ which detect, correspondingly, a stuck-at fault s_i /1 and a stuck-at fault s_i /0 at least at one input s_i, of the gate are sufficient for detecting all the design errors related to the replacement of AND by another gate and the invertor errors at all the inputs of the AND gate.

3. Consider now the case of design errors related to the OR gate. Let us choose a test pattern

$$T_{OR,1} = \{s_i = 1,\ \forall j,\ j \neq i: s_j = 0\},$$

which is one of solutions (2) of the equation (1), and which detects the stuck-at fault s_i/0 at the OR input. It is easy to see that the pattern $T_{OR,1}$ detects not only the error OR $\rightarrow$ AND, but also the error OR $\rightarrow$ NOR, and the errors of missing/extra invertors at the input s_i.

Consider now the error OR $\rightarrow$ NAND. The necessary condition for detecting this error can be formulated as:

$$(s_1 \vee s_2 \vee \ \ldots\ s_h) \oplus \neg\ (s_1 \wedge s_2 \wedge \ldots\ s_h) = 1. \tag{6}$$

There are two solutions for this equation, which can be formulated as (4) and (5). The solution (5) gives a test pattern

$$T_{OR,2} = \{\forall i,\ i = 1,2, \ldots h: s_i = 0\},$$

which detects not only the design error OR $\rightarrow$ NAND, but also stuck-at faults s_i /1 at all of the OR inputs. It is easy to see that the pattern $T_{OR,2}$ detects also all the errors of missing/extra invertor at the other OR gate inputs s_j, $j \neq i$ which were not detected by the pattern $T_{OR,1}$.

Hence, we have shown that the test patterns $T_{OR,1}$ and $T_{OR,2}$ which detect, correspondingly, a stuck-at fault s_i/0 and a stuck-at fault s_i/1 at least at one input of the gate are sufficient for detecting all the design errors related to the replacement of OR by another gate and all the invertor errors at the inputs of the OR gate.

4. In similar way as in points 1 and 2, we can show that the test patterns $T_{AND,1}$ and $T_{AND,2}$ which detect a stuck-at fault s_i/1 and a stuck-at fault s_i /0 at least at one input of the NAND gate, are sufficient for detecting all the replacements of a NAND by another gate, and the invertor errors at all the inputs of the NAND gate.

5. In the same way as in points 1 and 3, we can show that the test patterns $T_{OR,1}$ and $T_{OR,2}$ which detect a stuck-at fault s_i /0 and a stuck-at fault s_i /1 at least at one input of the NOR gate are sufficient for detecting all the replacements of a NOR by another gate, and the invertor errors at all the inputs of the NOR gate. ■

From the proof of the Theorem 1, the following set of corollaries follows which describe the mapping from a stuck-at fault diagnosis to a design error diagnosis.

Corollary 3.1. Localizing both the stuck-at-1 and stuck-at-0 faults on two or more gate inputs refers to the missing/extra invertor at the output of the gate, i.e. to the replacement errors: AND $\leftrightarrow$ NAND and OR $\leftrightarrow$ NOR.

Corollary 3.2. Localizing stuck-at-1 faults at one or more gate inputs refers to the replacement errors: AND $\rightarrow$ OR, OR $\rightarrow$ NAND, NAND $\rightarrow$ NOR, and NOR $\rightarrow$ AND.

Corollary 3.3. Localizing stuck-at-0 faults at one or more gate inputs refers to the replacement errors: AND $\rightarrow$ NOR, OR $\rightarrow$ AND, NAND $\rightarrow$ OR, and NOR $\rightarrow$ NAND.

Corollary 3.4. Localizing both the stuck-at-1 and stuck-at-0 faults at one of the gate inputs s_i refers to the error $s_i \rightarrow$ NOT(s_i) at this input.

From the single error hypothesis, the following statement in relation to extra/missing inverters errors results.

Corollary 3.5. Localizing both the stuck-at-1 and stuck-at-0 faults at more than one branch of a primary input $s_i \in X^F$ refers to the error $s_i \rightarrow$ NOT(s_i) at this input.

Example 3.1. As a direct illustration of Theorem 1 and Corollaries 3.1 - 3.4 , the mapping between localized stuck-at faults and design errors for 2-input gates is shown in Table 1.

Gate	Stuck-at faults				Design error	Gate	Stuck-at faults				Design error
	s_1		s_2				s_1		s_2		
AND	0	1	0	1	NAND	NAND	0	1	0	1	AND
		1		1	OR		0		0		OR
	0		0		NOR			1		1	NOR
	0	1			NOT(x_1)		0	1			NOT(x_1)
			0	1	NOT(x_2)				0	1	NOT(x_2)
OR	0	1	0	1	NOR	NOR	0	1	0	1	OR
	0		0		AND			1		1	AND
		1		1	NAND		0		0		NAND
	0	1			NOT(x_1)		0	1			NOT(x_1)
			0	1	NOT(x_2)				0	1	NOT(x_2)

Table 1: Mapping between stuck-at faults and gate errors

4. Test Generation for Detecting Design Errors with Structurally Synthesized BDDs

The test patterns generated for detecting the stuck-at faults in combinational circuits can be used for detecting simple gate design errors. We now consider a method which was developed for macro-level test generation based on using structurally synthesized BDDs (SSBDD) as the model for macros[7]. Test patterns are generated and faults are detected at the macro-level, however the fault (and error) diagnosis is made at the gate-level. Therefore, a correspondence should be established to map the macro-level results again back to the gate-level.

Consider a given implementation as a network of *macros* NF = $\{f_k\}$, where each macro is a tree-like subnetwork whose inputs $s \in S_k$ are either primary inputs which are not fanouts, $s \in X - X^F$, or branches of the fanout nodes of the network, $s \in Z^{FG}$. Each macro $f_k \in$ NF implements a function $s_k = f_k(s_k^1, s_k^2, ..., s_k^p)$, given in an equivalent parenthesis form (EPF) [8], where the arguments $s_k^j \in S_k$ in EPF are considered as literals.

Definition 4.1. Signal paths. Let $s_k = f_k(s_k^1, s_k^2, ..., s_k^p)$ be a macro implemented at the gate level, and $S_k = \{s_k^1, s_k^2, ..., s_k^p\}$ be its set of inputs. We denote $L(s_k^j)$ the set of variables on a path) from the input of the macro $s_k^j \in S_k$ to its output s_k. As macros are trees, there exists a one-to-one correspondence between inputs $s_k^j \in S_k$ and the gate-level signal paths $L(s_k^j)$ in the macro. The literal s_k^j in the EPF is an inverted (not inverted) variable if the number of invertors on the path from s_k^j to s_k is odd (even).

Definition 4.2. A SSBDD is a graph $G_k = (M_k, \Gamma_k, S_k)$ with a set of nodes M_k, which represents a macro f_k so that a one-to-one correspondence exists between the nodes $m \in M_k$ and signal paths $L(s)$ where $s \in S_k$. The set of nodes M_k is partitioned into nonterminal nodes M_k^N and terminal nodes M_k^T, $M_k = M_k^N \cup M_k^T$. There is one initial node $m_0 \in M_k^N$ and only two terminal nodes: $M_k^T = \{m^{T,0}, m^{T,1}\}$. The terminal nodes are labelled by constants 0 and 1, whereas the nodes $m \in M_k^N$ are labelled by literals $s \in S_k$. There is a mapping from the set of nodes of the SSBDD to the the set of literals of the EPF: let s(m) denote the literal at the node m. The mapping $\Gamma_k(m, e)$ defines the successor of m for the value of s(m) = e, $e \in \{0, 1\}$. Denote $\Gamma_k(m, e) = m^e$. A test pattern T_i which assigns values to S_k, defines a set of activated edges in G_k. The edge between m and m^e is activated when s(m) = e for the pattern T_i. Activated edges which connect nodes m_i and m_j make up an activated path in the graph (an ordered subset of nodes) $l(m_i, m_j) \subseteq M_k$. A path $l(m_0, m^{T,e})$ is called *fully activated path.* A SSBDD $G_k = (M_k, \Gamma_k, S_k)$ represents a gate-level network which implements the function $s_k = f_k(s_k^1, s_k^2, ..., s_k^p)$ iff for each pattern T_i, a full path $l(m_0, m^{T,e})$ in G_k will be activated such that $s_k = e$ [7,8]. The procedure of formal synthesis of SSBDDs from gate-level networks based on a graph superposition procedure is considered in [7,8].

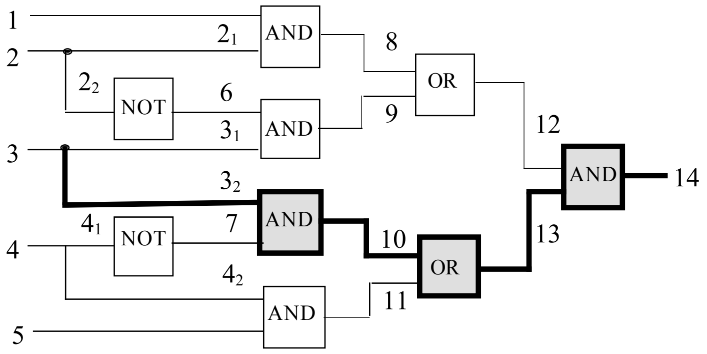

Fig.1. Combinational circuit

Example 4.1. Consider the combinational circuit of Fig.1 given as a Boolean function in an EPF as follows:

$$y = ((x_1 \wedge x_{2,1}) \vee (\neg x_{2,2} \wedge x_{3,1})) \wedge ((x_{3,2} \wedge \neg x_{4,1}) \vee (x_{4,2} \wedge x_5)).$$

The circuit in Fig.1 is represented by a structurally synthesized BDD in Fig.2. For simplicity, only indexes of the variables $s \in S$ at the nodes of the circuit and of the SSBDD are shown. The one-to-one correspondence between paths L in the circuit, literals x_i in the EPF, and nodes m in the SSBDD is given in Table 2. Table 2 illustrates the mapping from nodes m of the SSBDD both to the literals of EPF by s(m), and to the gate-level paths of the implementation by L(s(m)).

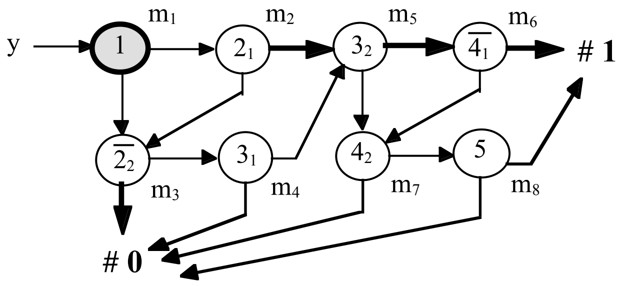

Fig.2. SSBDD for the combinational circuit in Fig.1.

Node in the SSBDD	m	m_1	m_2	m_3	m_4
Literal in the EPF	$s(m) \in S_k$	x_1	$x_{2,1}$	$\neg x_{2,2}$	$x_{3,1}$
Path in the circuit	$L(s(m))$	14, 12, 8, 1	14, 12, 8, 2_1	14, 12, 9, 6, 2_2	14, 12, 9, 3_1
Node in the SSBDD	m	m_5	m_6	m_7	m_8
Literal in the EPF	$s(m) \in S_k$	$x_{3,2}$	$\neg x_{4,1}$	$x_{4,2}$	x_5
Path in the circuit	$L(s(m))$	14, 13, 10, 3_2	14, 13, 10, 7, 4_1	14, 13, 11, 4_2	14, 13, 11, 5

Table 2: Correspondence between nodes, literals and paths

Theorem 4.1. A node $m \in M_k^N$ in the SSBDD G_k, is tested for a fault $s(m)/e$ by a test pattern T_i iff the test activates in the graph the following three paths: $l_1 = l(m_0, m)$, $l_2 = l(m^0, m^{T,0})$, $l_3 = l(m^1, m^{T,1})$, and $s(m) = \neg e$.

The proof is given in [7].

If $s_k \notin Y$, then a path in the circuit should be activated from s_k through other macros to some of the primary outputs of the network.

Theorem 4.2. If a test pair (T_1,T_2) which detects both stuck-at faults $s(m)/1$ and $s(m)/0$ at the node $m \in M_k^N$ in G_k, does not show an error, then all the gates along the path $L(s(m))$ in the gate-implementation are free from design errors.

Proof. From the definition of SSBDDs, it follows that a node m in G_k labelled by a variable $s(m)$ represents the signal path $L(s(m))$ in the circuit. In [7] it was shown that to test the faults $s(m)/1$ and $s(m)/0$ in G_k is equivalent to testing all the stuck-at faults along the path $L(s(m))$. In other words, if a test pair (T_1, T_2) which detects the stuck-at faults $s(m)/1$ and $s(m)/0$ at the node m in SSBDD G_k shows no error on the implementation outputs, it means that no stuck-at faults on the path $L(s(m))$ in the gate-level implementation can be present. In accordance with Theorem 3.1 and Corollaries 3.1 - 3.5, it also means that no design errors on the path $L(s(m))$ can be present. ■

Example 4.2. Let us create a test pair for testing both stuck-at faults $s(m_5)/0$ and $s(m_5)/1$ at the node m_5 in Fig.2. To test the fault $s(m_5)/0$ (which means $x_{3,2}/0$), we find the needed assignments according to Theorem 4.1: $l_1 = (m_1, m_2) \rightarrow \{ x_1 = 1, x_2 = 1\}$, $l_2 = (m_7, \#0) \rightarrow \{ x_4 = 0\}$, $l_3 = (m_6, \#1) \rightarrow \{ x_4 = 0\}$, and $s(m_5) = 1 \rightarrow x_3 = 1$, which results in the test pattern T_1 = (1110U). To test the fault $s(m_5)/1$ (which means $x_{3,2}/1$), we activate the same paths l_1, l_2, l_3, and assign $s(m_5) = 0 \rightarrow x_3 = 0$, which results in the second test pattern T_2 = (1100U). According to Theorem 4.2, these patterns detect all the stuck-at faults on the signal path $L(s(m_5)) = L(x_{3,2}) = \{3_2,10,13,14\}$ in the circuit in Fig.1. The tested path is highlighted in Fig.1 by bold lines

Corollary 4.1. If a test pattern T_i which shows an error detects a fault $s(m)/e$, $e \in \{0,1\}$ in f_k, then a stuck-at fault at each node along the signal path $L(s(m))$ is suspected as faulty, whereas the type of the suspected fault $e \in \{0,1\}$ at a node is e=1 (e=0) if the number of invertors from $s(m)$ to s_k is odd (even).

Example 4.3. Suppose a test pattern T_i = (11110) which detects the faults $\{s(m_6)/1 \rightarrow x_{4,1}/0,\ s(m_8)/1 \rightarrow x_5/1\}$ shows an error. Then the following gate-level stuck-at faults are suspected: $x_{4,1}/0 \rightarrow \{x_{4,1}/0, x_7/1, x_{10}/1, x_{13}/1, x_{14}/1\}$, and $x_5/1 \rightarrow \{x_{11}/1, x_{13}/1, x_{14}/1\}$.

5. Design Error Diagnosis Using Fault Tables for Stuck-at Faults

The test patterns generated for detecting the stuck-at faults in combinational circuits can be used for diagnosing single gate design errors. Assume we have generated a set of test patterns $T = \{T_1, T_2, \dots T_t\}$ for detecting the stuck-at faults at gate inputs and at the primary inputs of the circuit which are fanouts.

Definition 5.1. Detectable faults. Let us denote $F(T_i, y)$ the set of stuck-at faults detectable by the test pattern T_i at the primary output $y \in Y$. For each one of these faults, s_j/e, the value of variable s_j should be different from e under T_i and this fault is propagated to the output y. Mathematically:

$$F(T_i, y) = \{s_j/e \mid s_j \in S,\ T_i \rightarrow (\partial y/\partial s_j = 1) \ \&\ (s_j = \neg e)\} \subseteq F$$

We denote E (T_i) the subset of primary outputs where an error has been detected by applying the test pattern T_i.

$$E(T_i) = \{y_k \in Y \mid y_k(T_i) \neq w_k(T_i)\} \subseteq Y.$$

Theorem 5.1. If a test pattern T_i shows an error, the following set of suspected faults results

$$F(T_i) = \cap_{y \in E(Ti)} F(T_i, y) - \cup_{y \in Y - E(Ti)} F(T_i, y).$$

$F(T_i)$ is the set difference between (1) the intersection of the sets of faults detectable at the primary outputs where an error was shown and (2) all the sets of faults detectable at the primary outputs where no error was shown.

Proof. The proof results from the single error hypothesis. If an error has been detected at more than one output $y \in E(T_i) \subseteq Y$ then only a single fault can be the cause of that. Therefore, only the intersection of the sets of suspected faults $F(T_i, y)$ at erroneous outputs $y \in E(T_i)$ can contain the existing fault. On the other hand, if some suspected faults from this intersection have a direct impact on the outputs where no error has been detected, they can no longer be suspected. Therefore, the union of all $F(T_i, y)$ for all $y \in Y - E(T_i)$ should be substracted from the intersection of suspected faults observed at erroneous outputs $y \in E(T_i)$. ■

Corollary 5.1. For a diagnostic experiment with a test T, let $E \subseteq T$ be the subset of test patterns that show an error; the set of suspected faults F(T) can be calculated in the following way:

$$F(T) = \cap_{T_i \in E} F(T_i) - \cup_{T_i \in T - E} F(T_i).$$

T_i	Test pattern	Fault Table (detected faults at variabes $s(m_i)$)								E
		$s(m_1)$ x_1	$s(m_2)$ $x_{2,1}$	$s(m_3)$ $\neg x_{2,2}$	$s(m_4)$ $x_{3,1}$	$s(m_5)$ $x_{3,2}$	$s(m_6)$ $\neg x_{4,1}$	$s(m_7)$ $x_{4,2}$	$s(m_8)$ x_5	
T_1	1110x	0	0			0	0			0
T_2	0110x	1		1						1
T_3	11011	0	0					0	0	0
T_4	10011		1		1					1
T_5	0010x			0	0	0	0			0
T_6	11001					1		1		0
T_7	11110						1		1	0
T_8	00011				1					0

Table 3. Stuck-at fault table

Example 5.1. Consider a test experiment $T = \{ T_1, T_2 \ldots T_7 \}$ with 7 test patterns which are applied to the inputs of the circuit in Fig.1 and Fig.2. The test patterns $T_i \in T$ and the sets of detectable faults $F(T_i, y_k)$ are described in Table 3. The entry "1" or "0" in a columns $s(m_i)$ means detection of the fault $s(m_i)/1$ or $s(m_i)/0$. As we see, the test is complete, all the stuck-at faults in the circuit are tested, and according to Theorem 4.2. this test is able to detect all single gate design errors. Now suppose the test patterns T_2 and T_4 show an error. According to Theorem 5.1, we have $F(T_2) = \{x_1/1, x_{2,2}/0\}$, and $F(T_4) = \{x_{2,1}/1, x_{3,1}/1\}$. In this case, Corollary 5.1 cannot reduce this set of suspected faults, hence: $F(T) = \{x_1/1, x_{2,2}/0, x_{2,1}/1, x_{3,1}/1\}$. From Corollaries 4.1, and Theorem 3.1, the suspected set of erroneous gates determined by test T is: Suspected(T) = $\{g_6, g_8, g_9, g_{12}\}$. By adding an additional test T_8 = (00011) which detects the fault $x_{3,1}/1$, we can remove the gates g_9 and g_{12} from this list. As $x_{2,2}/0$ is suspected, but $x_{2,2}/1$ is correct then, according to Corollary 3.4, the gate g_6 is also correct. From the final list Suspected (T) = $\{g_8\}$ and from Corollary 3.2, it follows that in the circuit a design error $AND_8 \rightarrow OR_8$ is present.

6. Considerations on the Efficiency of the Approach and Experimental Data

The proposed diagnostic procedure consists of two parts : error detection and error diagnosis. Both parts are based on using the stuck-at fault model and the hierarchical representation of random logic by SSBDDs. The complexity of test generation (the number of tests needed) for error detection is determined by the possibility of covering all the gates of the circuit by as few paths as possible.

The number of generated and compacted test patterns needed for fault detection is given in Table for ISCAS'85 circuits. The efficiency of test generation when using SSBDDs for representing macros instead of gates is illustrated by the time (in seconds) needed for test generation. The last column illustrates the efficiency of using SSBDDs when determining the faults detected by test patterns as the basic operation in fault diagnosis. A ratio of simulation speeds between using the gate level and macro level is given in the last column.

ISCAS circuit	Number of gates	Fault cover, %	Number of test patterns	Compacted patterns	TPG time, s	Fault simulation gate/macro speed ratio
c432	160	97.33	89	55	0.10	3,86
c880	383	100.00	140	100	0.05	5,15
c1355	546	99.64	70	52	0.24	3,08
c1908	880	99.75	144	122	0.22	6,46
c2670	1193	96.67	160	119	0.55	6,47
c3540	1669	95.58	201	145	0.77	8,81
c5315	2307	99.78	178	108	0.57	8,37
c6288	2416	99.80	41	33	0.60	2,55
c7552		99.46	276	198	2.71	9,04

Table 4. Experimental data

For fault diagnosis the method proposed in the paper has the following advantages compared to the previous work [1]:

1. The design error detection and localization are combined and based on the same technique; this facilitates the use of the information about error free nodes, already obtained during the error detection procedure, for error localization.

2. The whole procedure takes place hierarchically at three different levels: macro level (for error detection and for localization of the erroneous macro), gate level (for localization of the node related to the site of the design error), and "stuck-at fault to design error mapping" level for exact specification of the design error. Exploiting the hierarchy allows to combine the efficiency of working at the higher level (for error detection) with the accuracy (needed for error diagnosis) at the lower level.

3. Working with the stuck-at fault model, on a single error hypothesis, corresponds to working with all three hypothesis from [1] in parallel.

7. Conclusions.

In this paper, a new approach has been presented, to automatically diagnose single design errors in combinational circuits. The main original features of the method are: the hierarchical approach, based on using structurally synthesized BDDs, the use of very powerful error detection and fault localization procedures based on SSBDDs and the idea of mapping stuck-at fault diagnosis into the final localization of the design error. The latter allows to use the test patterns and fault tables generated for stuck-at faults to produce design error diagnosis. Experimental data are provided for showing the efficiency of the error detection phase of the method. The efficiency of the second phase - error site localization, results from the drastically reduced area where the searh for the faulty gate should be continued after error detection. The work on showing the efficiency of the error localization with support of corresponding experiments has been started. The future research in this field is directed to the case of multiple design errors and to the case of complex gates. The use of word level DDs seems to be very efficient in design error diagnosis at higher functional levels like RTL or behavioral ones.

Acknowledgements. Authors appreciate the work of Jaan Raik, Gert Jervan and Antti Markus from the Tallinn Technical University for carrying out the experiments.

References

1. A.M. Wahba, D. Borrione. A Method for Automatic Design Error Location and Correction in Combinational Logic Circuits. *J. of Electronic Testing: Theory and Applications* 8, 113-127 (1996).
2. A.M. Wahba. *Diagnostic des Erreurs de Conception dans les Circuits Digitaux: le Cas des Erreurs Simples*. PhD Dissertation. UJF/TIMA, Grenoble, 1997, 156 p. (In French)
3. K.A. Tamura. Locating Functional Errors in Logic Circuits. *Proc. 26th Design Automation Conf.*, 1989, pp. 185-191.
4. J.C. Madre, O.Coudert, J.P.Billon. Automating the Diagnosis and the Rectification of Design Errors with PRIAM. *Proc. ICCAD'89*, 1989, pp.30-33.
5. M. Tomita, T.Yamamoto, F.Sumikawa, K.Hirano. Rectification of Multiple Logic Design Errors in Multiple Output Circuits. *Proc. 31st Design Automation Conf.*, 1994, pp. 212-217.
6. P.Y. Chung, Y.M. Wang, I.N. Hajj. Diagnosis and Correction of Logic Design Errors in Digital Circuits. *Proc. 30th Design Automation Conf.*, 1993, pp. 503-508.
7. R. Ubar. Test Synthesis with Alternative Graphs (R.Ubar). *IEEE Design and Test of Computers.* Spring, 1996, pp.48-59.
8. R. Ubar. Combining Functional and Structural Approaches in Test Generation for Digital Systems. *Micro-electronics and Reliability*. Elsevier Science Ltd. No.1, pp.1-13, 1998.