HIERARCHICAL DESIGN ERROR DIAGNOSIS IN COMBINATIONAL CIRCUITS BY STUCK-AT FAULT TEST PATTERNS

R.UBAR, A.JUTMAN
TALLINN TECHNICAL UNIVERSITY, ESTONIA

KEYWORDS: Design Errors, Stuck-at Faults, Fault Localization, Combinational Circuits, Decision Diagrams

ABSTRACT: A new hierarchical design error diagnosis algorithm for combinational circuits is proposed, which is based on the stuck-at fault model and assumes the case of single logic gate errors. Decision diagrams are used for representing and localizing stuck-at faults at the higher signal path level. On the basis of detected faulty signal paths, suspected stuck-at faults at gate inputs are calculated, and then mapped into suspected design error(s). Using the stuck at fault model allows to exploit standard gate-level automated test pattern generators (ATPG) for design error diagnosis. Experimental data on wellknown benchmark circuits show the advantage of the proposed method compared to the known algorithms of design error diagnosis.

INTRODUCTION

Design verification and design error localization in digital systems are becoming more and more time consuming tasks due to the fact that systems are becoming continuously more complex.

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 [7,8] for solving the same problem as in [1], a new model of structurally synthesized BDDs (SSBDD) [9] was introduced which allowed to increase the speed in error detection and localization. Differently from [1] where only the diagnosis problem is formulated and solved, in [7,8] the error detection and diagnosis are handled as a joint task which allows to increase the efficiency of error localization.

In this paper a new algorithm for hierarchical design error diagnosis is presented which is based on general ideas presented in [7,8], and in details describes the fault calculation procedure.

The use of SSBDDs allows to develop efficient higher than gate level fault reasoning procedures for increasing the speed in fault simulation and fault diagnosis. On SSBDDs, a primary set of suspected faulty signal paths are calculated. Based on these paths, a list of suspected faults is generated, which subsequently will be reduced to the minimum by using the information obtained from the test experiment.

The method 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 are explained in Section 3, and representation of faults in the model is discussed in Section 4. The new error diagnosis technique is presented in Section 5. Experimental data are discussed in Section 6, and finally, Section 7 presents some conclusions.

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 $W = \{w_1, w_2, \dots, w_m\}$, and the implementation output is given by a set of variables $Y = \{y_1, y_2, \dots, y_m\}$, where m is the number of outputs. Let $X = \{x_1, x_2, \dots, x_n\}$ 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 implement simple Boolean functions AND, NAND, OR, 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: 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. At the gate level, the network is described by a set NG = $\{g_k\}$ of gate functions $s_k = g_k (s_k^1, s_k^2, ... ,s_k^h)$ 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. Then, at the macro-level, the network is given by a set NF = $\{f_k\}$ of macro functions $s_k = f_k (s_k^1, s_k^2, ..., s_k^p)$ where $s_k \in Y \cup Z^F$, and $s_k^j \in Z^{FG} \cup (X - X^F)$.

<u>*Definition 1.*</u> *Test patterns.* For a circuit with n inputs, a *test pattern* 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.

<u>*Definition 2.*</u> *Stuck-at fault set.* Let F be the set of stuck-at faults s/1 and s/0, where $s \in Z \cup X$.

<u>*Definition 3.*</u> *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.

<u>*Definition 4.*</u> *Stuck-at fault cover.* The circuit is tested completely by a test $T = \{T_1, T_2, ... T_t\}$ for stuck-at faults, iff T detects all the faults in F. The gate g_k which implements the function $s_k = g_k (s_k^1, s_k^2, ... ,s_k^h)$ 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 $g_k \in NG$.

<u>*Definition 5.*</u> *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 $g_i \rightarrow g_j$.

<u>*Definition 6.*</u> *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 $s \in Z^{FG}$: $s \rightarrow NOT(s)$.

<u>*Definition 7.*</u> *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.

MAPPING GATE STUCK-AT FAULTS INTO DESIGN ERRORS

<u>*Theorem 1.*</u> To detect a design error in the implementation at an arbitrary gate g_k where $s_k = g_k (s_1, s_2,...,s_h)$, 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, ... h.

The proof of the Theorem 1 is given in [7]. From the proof the following set of corollaries was driven which describes the mapping from a stuck-at fault diagnosis to a design error diagnosis:
- localizing both the s/1 and s/0 faults on two or more gate inputs refers to the missing/extra invertor at the gate output, i.e. to the replacement errors: AND $\leftrightarrow$ NAND and OR $\leftrightarrow$ NOR;
- localizing s/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;
- localizing s/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;
- localizing both the s/1 and s/0 faults at one of the gate inputs s_i refers to the error $s_i \rightarrow NOT(s_i)$ at this input;
- localizing both the s-1 and s-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.

The mapping of stuck-at faults into simple gate design errors can be represented by the following table.

TABLE 1

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

MODELING STUCK-AT FAULTS AT TWO DIFFERENT LEVELS

We now consider a fault modeling method which was developed for macro-level test generation based on using structurally synthesized BDDs (SSBDD) as the model for tree-like subcircuits or macros [9,10].

Consider a given implementation as a network of *macros* (tree-like subcircuits) NF=$\{f_k\}$, where each macro implements a function $s_k = f_k (s_k^1, s_k^2, ..., s_k^p)$, given in an equivalent parenthesis form (EPF) [10], where the arguments $s_k^j \in S_k$ in the EPF are considered as literals.

Definition 8. *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 9. *Faults on signal paths.* Let $F(s_k^j/e)$ where $e \in \{0,1\}$ be a set of all faults (*a fault class*) in the gate-level signal path from the input s_k^j of the macro S_k to its output s_k.

A fault s_k^j/e can be regarded as the representative of the fault class $F(s_k^j/e)$, since to test all the faults $F(s_k^j/e)$ it is enough to test only the fault s_k^j/e.

A SSBDD is a graph G_k with a set of nodes M_k, which represents a macro f_k so that one-to-one correspondence exists between the nodes $m \in M_k$ and signal paths $L(s)$ where $s \in S_k$ [9,10]. Let $s(m)$ denote the literal at the node m in the graph G_k.

The procedure of formal synthesis of SSBDDs from gate-level networks based on a graph superposition procedure is considered in [7,8].

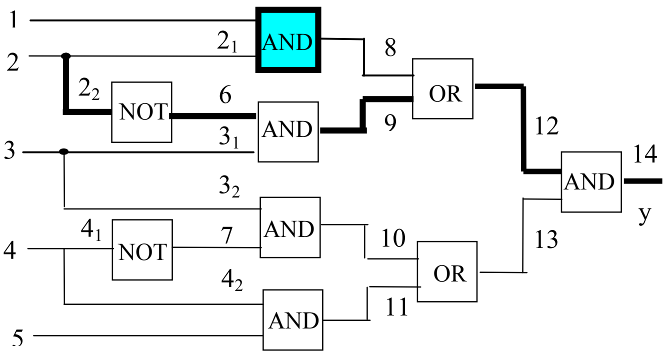

Fig.1. Combinational circuit

Example 1. Consider a combinational circuit in 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.

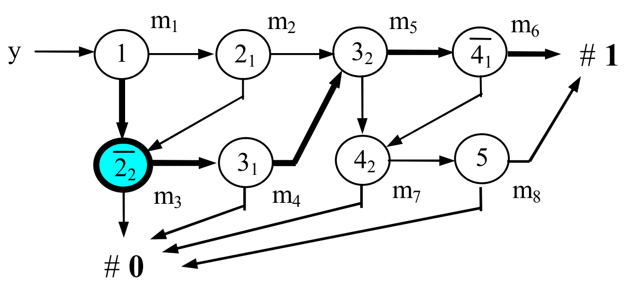

Fig.2. SSBDD for the combinational circuit in Fig.1.

The one-to-one correspondence between the fault classes $F(s(m)/e)$, $e \in \{0,1\}$, on paths $L(s(m))$ in the circuit, the representative faults $s(m)/e$, and nodes m_i in the SSBDD, is given in Table 2.

As an example, to test the node m_3 in the SSBDD for a fault $\neg x_{2,2}/0$ (for stuck-at 1 at the branch $x_{2,2}$ in Fig.1) we activate the paths shown by bold arrows in SSBDD (Fig.2) by the pattern x_1=0,x_2=0,x_3=1,x_4=0. This pattern detects all the faults $x_{2,2}/1$, $x_6/0$, $x_9/0$, $x_{12}/0$, $x_{14}/0$ on the signal path (bold in Fig.1) from the input branch $x_{2,2}$ to the output y of the circuit . Note the change of the fault type because of the invertor on the path.

TABLE 2

Nodes of the SSBDD m_i	Faults in the SSBDD $s(m)/e$	Faults on the gate level $F(s(m)/e)$, $e \in \{0,1\}$
m_1	x_1/e	x_1/e, x_8/e, x_{12}/e, x_{14}/e
m_2	$x_{2,1}/e$	$x_{2,1}/e$, x_8/e, x_{12}/e, x_{14}/e
m_3	$\neg x_{2,2}/e$	$x_{2,2}/\neg e$, x_6/e, x_9/e, x_{12}/e, x_{14}/e
m_4	$x_{3,1}/e$	$x_{3,1}/e$, x_9/e, x_{12}/e, x_{14}/e
m_5	$x_{3,2}/e$	$x_{3,2}/e$, x_{10}/e, x_{13}/e, x_{14}/e
m_6	$\neg x_{4,1}/e$	$x_{4,1}/\neg e$, x_7/e, x_{10}/e, x_{13}/e, x_{14}/e
m_7	$x_{4,2}/e$	$x_{4,2}/e$, x_{11}/e, x_{13}/e, x_{14}/e
m_8	x_5/e	x_5/e, x_{11}/e, x_{13}/e, x_{14}/e

DESIGN ERROR DIAGNOSIS AT PATH AND GATE LEVELS

The test patterns generated by traditional gate-level ATPGs 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 $T = \{T_1, T_2, ... T_t\}$ for detecting the stuck-at faults at gate inputs and at the primary inputs of the circuit which are fanouts.

Definition 10. *Detectable faults.* Let us call

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

where $s_j \in S$, a set of faults detectable by a test pattern T_i at a primary output $y \in Y$.

Let us call

$$F(T_i) = \cup_{y \in Y} F(T_i, y)$$

a set of faults detectable by a test pattern T_i.

Denote the subset of primary outputs where an error has been detected by applying the test pattern T_i as

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

Theorem 2. From failing of a test T_i, the following set of suspected faults results

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

Proof. The proof results from the single error hypothesis. If an error has been detected at more than one outputs $y \in E(T_i) \subseteq Y$ then only a single fault can be the cause of that. Therefore, only the intersection of sets of suspected faults $F(T_i, y)$ at erroneos outputs $y \in E(T_i)$ can contain the existing fault. On the other hand, if some of these suspected faults from this intersection have a direct impact to the outputs where no error has been detected, they cannot be anymore suspected. Therefore, the union of all $F(T_i, y)$ for all $y \in Y\text{-}E(T_i)$ should be substracted from the intersection of suspected faults observed at erroneos outputs $y \in E(T_i)$. ν

Theorem 3. From a set E of failing test patterns where

$$E = \{ T_i \mid \exists j: T_i \rightarrow (\partial y/\partial s_j = 1) \} \subseteq T,$$

the following set of suspected faults results

$$F^*(E) = \cup_{T_i \in E} [\cap_{y \in E(T_i)} F(T_i, y)] -$$

$$\cup_{T_i \in E} [\cup_{y \in Y\text{-}E(T_i)} F(T_i, y)] - \cup_{T_i \in F\text{-}E} F(T_i)$$

Proof. The proof results again from the single error hypothesis. From all the failing test patterns $T_i \in E$, a suspected set

$$F'(E) = \cup_{T_i \in E} [\cap_{y \in E(T_i)} F(T_i, y)]$$

of faults results.

On the other hand, since all the faults

$$F''(E) = \cup_{T_i \in E} [\cup_{y \in Y\text{-}E(T_i)} F(T_i, y)],$$

not detected by test patterns $T_i \in E$, and all the faults

$$F''(F\text{-}E) = \cup_{T_i \in F\text{-}E} F(T_i)$$

not detected by test patterns $T_i \in F\text{-}E$, cannot be suspected, we have to substract $F''(E)$ and $F''(F\text{-}E)$ from $F'(E)$. ν

From Theorems 2 and 3 the following algorithm for fault diagnosis results.

Algoritm 1.

1. Calculate $F'(E) = \cup_{T_i \in E} [\cap_{y \in E(T_i)} F(T_i, y)]$ as a set of suspected faults.
2. Calculate $F''(E) = \cup_{T_i \in E} [\cup_{y \in Y\text{-}E(T_i)} F(T_i, y)]$ as a subset of not suspected faults.
3. Calculate $F''(F\text{-}E) = \cup_{T_i \in F\text{-}E} F(T_i)$ as another subset of not suspected faults.
4. Calculate $F^*(E) = F'(E) - F''(E) - F''(F\text{-}E)$ as the updated set of suspected faults.

For improving the resolution of diagnosis, the following theorem can be used.

Theorem 4. If two test patterns T_1 and T_2 which detect the both stuck-at faults s(m)/1 and s(m)/0 at the node m in G_k, will not fail then all the gates along the path L(s(m)) in the gate-implementation are free from design errors.

The proof of the theorem is given in [7].

Example 2. Consider a test T with 8 patterns which are applied to inputs of the circuit in Fig.1 described by SSBDD in 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 entries "1" or "0" in a columns i for $s(m_i)$ mean, correspondingly, 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 this test is able to detect all single gate design errors.

TABLE 3

T_i	Test pattern	Faults detected at i: $s(m_i)$								E
		1	2	3	4	5	6	7	8	
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

Suppose now, the test patterns T_2 and T_4 fail which results in $E=\{T_2,T_4\}$. From that, according to Algorithm 1, we create an initial subset of suspected representative faults (high level diagnosis):

$$F'(E) = F(T_2) \cup F(T_4) = \{x_1/1, x_{2,2}/0, x_{2,1}/1, x_{3,1}/1\}.$$

On the other side, since $F''(E) = \varnothing$, the final set of suspected representative faults is:

$$F^*(E) = F'(E) - F''(F\text{-}E) = \{x_1/1, x_{2,1}/1, x_{2,2}/0\}.$$

Going now over to the lower level fault representation, we create from F*(E) according to Table 2, the following initial subset of suspected gate-level stuck-at faults (low level diagnosis):

$$F'(E) = \{x_1/1, x_{2,1}/1, x_{2,2}/0, x_6/1, x_8/1, x_{12}/1, x_{14}/1\}.$$

Using again Algorithm 1 we reduce the suspected fault set to as follows:

$$F^*(E) = F'(E) - F''(F\text{-}E) = \{x_1/1, x_{2,1}/1, x_{2,2}/0, x_6/1, x_8/1\}.$$

Since the faults at $x_{3,1}$ are missing, then according to Theorem 4, the gates g_9 and g_{12} are fault free, and correspondingly, the faults $x_6/1$ and $x_8/1$ can not be any more suspected. Hence, we remove them from F*(E).

Since the fault $x_{2,2}/0$, according to the mapping rules in Table 1, does not fit to any fault combination for the NOT gate, the gate g_6 is fault free, and we can remove also this fault from F*(E).

From above, the final stuck-at fault diagnosis results:

$$F^*(E) = \{x_1/1, x_{2,1}/1\},$$

which according to Table 1 means the design error $AND_8 \rightarrow OR_8$. To correct the design, the gate AND_8 (shaded in Fig. 1) should be replaced by gate OR.

EXPERIMENTAL DATA

For carrying out a set of design error diagnosis experiments, internationally recognized ISCAS´85 benchmarks (columns 1,2,3,4 in Table 4) were used. Test patterns for detecting stuck-at faults in these circuits were created by the test generator described in [10]. The fault coverage (column 6) of the tests and the test generation time in seconds (column 11) are presented in Table 4.

The goals of the experiments were twofold:

- to compare the efficiency (the speed of fault localization) of the new diagnostic approach in comparison with previous results [1,2];

TABLE 4

Circuit Name	Number of				Fault Coverage, %	Suspected Erroneous Gates				Time,s			Time for [1], s
	Inputs	Outputs	Gates	Experi-ments		Number			Av. % of Total	Test Gene-ration	Fault Analysis (average)	Total	
						Min	Max	Av.					
1	2	3	4	5	6	7	8	9	10	11	12	13	14
c432	36	7	232	671	91,07	1*	107	8,9	3,82	0,79	0,1	0,9	17,57
c499	41	32	618	1622	99,33	1	307	76,5	12,38	1,01	1,4	2,4	111,64
c880	60	26	357	1144	100	1	33	6,2	1,73	0,19	0,5	0,7	126,79
c1355	41	32	514	1830	99,51	1	248	58,4	11,37	1,35	1,5	2,9	241,79
c1908	33	25	718	1922	99,31	1	76	11,1	1,55	0,93	1,6	2,5	341,92
c2670	233	140	997	997*	94,97	1*	161	25,4	2,55	3,55	14,1	17,7	661,91
c3540	50	22	1446	1446*	95,27	1*	86	10,1	0,70	3,08	3,7	6,8	1513,82
c5315	178	123	1994	1994*	98,69	1*	239	11,1	0,56	2,38	29,4	31,8	1814,04
c6288	32	32	2416	2416*	99,34	1*	138	8,4	0,35	2,17	2,7	4,9	1895,90
c7552	207	108	2978	2978*	95,95	1*	269	16,0	0,54	12,06	44,8	56,9	

- to evaluate the design error diagnostic properties of test patterns generated by traditional gate-level ATPGs for only stuck-at fault detecting purposes.

Experiments were carried out on the computer platform Sun SparcServer 20 (2 x Ultra Sparc II micro-processors, 75MHz) with Solaris 2.5.1 OS.

The number of experiments carried out for each circuit are shown in the column 5. For the cases marked by star (*), one random error for each gate was inserted, for other cases, all possible single gate errors were simulated and analysed.

TABLE 5

Circuit Name	Level of Abstraction	Total	Suspected Erroneous Area			
			Number			Av. % of Total
			Min	Max	Av.	
c499	Nodes	601	2	365	93,0	15,47
	Gates	618	1	307	76,5	12,38
c1908	Nodes	866	1	131	17,5	2,02
	Gates	718	1	76	11,1	1,55

The efficiency in the speed of diagnosis (columns 11, 12, 13) are compared to the results of [1,2] (column 14). The total time of diagnosis (column 13) in this paper consists of two components: test generation time (column 11) and fault diagnosis (column 12).

However, it should be noted that in this paper the diagnostic resolution can not be compared with the one in [1,2] because the test patterns were originally not generated for diagnostic purposes. The numbers in columns 7, 8, and 9 show, correspondingly, the minimum, maximum and average diagnostic resolutions (numbers of suspected gates) reached by the tests.

In the cases marked by star (*) in column 7, some design errors were not detected at all (as the test patterns were not generated with dignosis purposes).

In Table 5, the diagnostic resolutions for two benchmark circuits (with the best and worst diagnostic resolutions) are shown for both higher and lower representation levels. The diagrams in Fig. 3 and 4 show, correspondingly, the distribution of experiments with different diagnostic resolutions (the best case for the circuit c1908, and the worst case for the circuit c499).

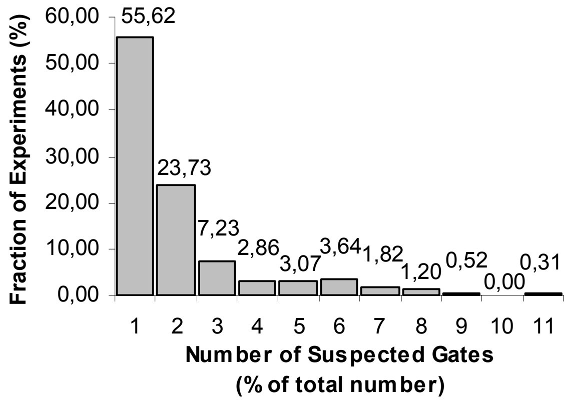

Fig.3. Distribution of diagnostic resolutions over all possible error cases for the circuit c1908.

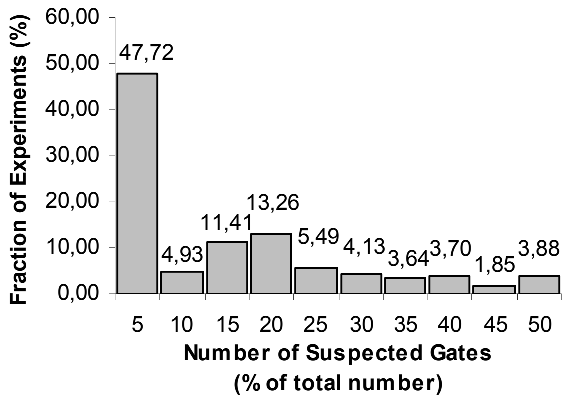

Fig.4. Distribution of diagnostic resolutions over all possible error cases for the circuit c499.

To reach the same high resolution of [1,2], additional test patterns should be generated. For this purpose, the same method of [1,2] can be used. Since the suspected area (column 10) for diagnostic search is reduced from 100% to from 0,35% (in the best case) to 12,38% (in the worse case), the combination of both methods – the one proposed in the present paper and the other one presented in [1,2] can reach significant improvements compared to using only the method [1,2].

A very interesting result is that in 30% cases (c499, c1355, c1908) the tests with lower than 100% stuck-at fault coverage detected all possible single gate design errors. The reason lies in the mapping mechanism explained in Table 1 where each design error is mapped into a subset of at least two stuck-at faults.

The method proposed in the paper has the following advantages compared to the previous results[1,2]:

1. The whole procedure takes place hierarchically at three different levels: macro level (for error detection and for localization of the erroneos 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 detecting) with the accuracy (needed for error diagnosis) at the lower level.

2. Working with the stuck-at fault model allows to base on a single error hypothesis, which actually means working with all the three hypothesis from [1,2] in parallel.

CONCLUSIONS

In this paper, a new approach to diagnosis of design errors is 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 method.

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 – This work has been supported by the COPERNICUS project No 977133 VILAB and by the Estonian Science Foundation grant G-1850. Authors appreciate the work of Jaan Raik for helping to carry out the experimental work.

THE AUTHORS

Prof. Raimund Ubar and Artur Jutman are with the Computer Engineering Department of Tallinn Technical University, Raja 15, 12617 Tallinn, ESTONIA. E-mail: raiub@pld.ttu.ee.

REFERENCES

[1] A.M. Wahba, D. Borrione. A Method for Automatic Design Error Location and Correction in Combinational Logic Circuits. Journal of Electronic Testing: Theory and Applications 8, 1996, pp. 113-127.

[2] A.M. Wahba. Diagnostic des Erreurs de Coception dans les Circuits Digitaux: le Cas des Erreurs Simples. The Thesis of PhD Dissertation. UJF/TIMA, Grenoble, 1997, 156 p.

[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, D.Borrione. Localization of Single Gate Design Errors in Combinational Circuits by Diagnostic Information about Stuck-at Faults. Proc. of the 2nd Int. Workshop on Design and Diagnostics of Electronic Circuits and Systems. Szczyrk, Poland, Sept. 2-4, 1998, pp.73-79.

[8] R.Ubar, D.Borrione. Generation of Tests for Localization of Single Gate Design Errors in Combinational Circuits Using the Stuck-at Fault Model. Proc. of the 11th IEEE Brasilian Symp. on IC Design. Rio de Janeiro, Brazil, Sept. 30 - Oct. 3, 1998, pp.51-54.

[9] R. Ubar. Test Synthesis with Alternative Graphs (R.Ubar). IEEE Design and Test of Computers. Spring, 1996, pp.48-59.

[10] J.Raik, R.Ubar. Feasibility of Structurally Synthesized BDD Models for Test Generation. Proc. of the IEEE European Test Workshop, Barcelona, May 27-29, 1998, pp.145-146.