Generation of Hazard Identification Functions

Maria K. Michael
CSE Dept., University of Notre Dame
maria@cse.nd.edu

Spyros Tragoudas
ECE Dept., Southern Illinois University
spyros@engr.siu.edu

Abstract

We study the problem of identifying the complete set of pairs of input patterns that can cause different types of hazards to appear at a circuit line. A novel methodology to implicitly identify all possible input configurations is proposed. The technique is based on a systematic derivation of the conditions for the occurrence of static and dynamic hazards at a line, which are subsequently formulated as Boolean functions defined over variables representing the primary input signals. Our experimental results demonstrate that the proposed approach is very promising and outperforms existing approaches. In addition, they show that a proposed solution for the decision problem of hazard existence at a circuit line is very efficient.

1 Introduction

Identification of hazards is a major problem encountered in many areas of VLSI CAD, such as circuit synthesis, timing analysis and verification, and testing. Hazards are of significant importance in asynchronous circuit design. In the context of combinational circuit synthesis, the existence of hazards may be of no consequence, as in the case of combinational modules embedded in synchronous systems. However, in the case where combinational logic generates signals that control internal variables of sequential circuits, the existence of hazards may lead to significant circuit malfunction. Moreover, combinational hazards must be considered in timing analysis and verification [9], [17], as well as test generation [4], [12]. In particular, in the context of testing, hazard identification becomes crucial in generating tests that will not be invalidated when applied to the circuit under test.

Combinational hazards, classified as either *static* or *dynamic*, have long received wide attention in the literature and have been attacked from a variety of directions, including hazard analysis [23], design of hazard-free circuits via hazard elimination [11] and [16], algebras for accurate hazard identification during vector simulation [6], and hazard-free test generation [1], [4], and [12].

Most of the original work on combinational hazards focused on the detection and elimination of hazards caused by single-input changes (SIC) (see [8] and [13], among others). The extension to multi-input change (MIC) transitions, which overall resulted in the identification of static hazards, was proposed in [7], which further classified MIC static hazards into *function* and *logic* hazards. Function static hazards are caused by stray delays present at the circuit inputs and can be detected by examining the boolean functions realized by the circuit. Thus, they are independent of a specific circuit implementation. Logic static hazards are closely related to internal propagation times on the signal paths and, thus, they depend on the circuit implementation.

[2] and [3], among others, extended earlier work to handle both static and dynamic hazard detection. The technique in [2] is particularly interesting since it is based on the definition of a unateness criterion for the appearance of static and dynamic hazards. However, the method depends on circuit unfolding and ultimately on path enumeration. A major disadvantage among most of the methods mentioned above is that they assume sum-of-products function representations, which is prohibitive for large circuits. Most importantly, a set of input changes (pairs of input patterns) is an input parameter to the detection problem. Boolean differential calculus ([21]), which is a comprehensive theoretical framework that encompasses and generalizes the algebraic concepts introduced by the former researchers, was applied to hazard detection in [20]. Separate criteria for static, dynamic, and even sequential hazards are presented in [20]. Unfortunately, this approach requires rather extensive computational work and cannot be effectively applied to large circuits. The possibility for an alternative methodology for static and dynamic hazard analysis is mentioned in the more recent work of [14], where a framework for generating path-recursive functions that can be used for timing analysis and delay test generations is presented. This is an overly complex framework for our problem formulation.

In this paper, we examine the problem of combinational hazard detection and propose a methodology to implicitly identify the complete set of input changes that could excite different types of combinational hazards at a circuit line. Thus, we examine the problem of generating boolean functions, one per circuit line, whose minterms correspond to all possible pairs of input patterns that can cause the existence of a particular type of hazard at the line. We refer to these functions as the *hazard identification* functions. The computed functions are defined over variables that represent the initial and final values of the transient primary input signals. The proposed approach is based

0-7695-1881-8/03 $17.00

on a systematic visiting of the circuit lines. At each line, the conditions for the occurrence of hazards are derived and formulated as a boolean function in a recursive manner with respect to the already derived hazard functions at the predecessor lines. Different functions are generated to distinguish between static and dynamic hazards. Under the above arguments, the proposed formulation for hazard identification functions does not depend on path recursive arguments that are required for the timing analysis and path delay fault generation procedures presented in [14].

Hazardous line behavior using functional sensitivity analysis is also studied in [10]. The concept of line stability functions from [1] is used and expanded to generate *glitchy signal transition sets*, that are used to predict signal interaction during the process of crosstalk identification in deep sub-micron designs. The glitchy signal transition sets are an alternative formulation for the hazard functions that we propose, and is based on the premise of identifying sets of input patterns that set a line to a stable value (stable-0, stable-1 or dynamic hazard-free transitions) and utilizing these sets to derive the hazardous input sets. Stability functions can be generated relatively fast and efficiently in many cases [15]. However, there are cases where the memory requirements are prohibitive (our experimental results in Section 4 verify this). Instead, we identify the necessary conditions that *must* occur at a gate's inputs in order to excite a hazard at the gate's output. This formulation tends to be more complex than the one in [10], but it specifically avoids the generation of stability functions.

In this work we assume that gates and wires can have arbitrary finite delays, which means that the circuit operates correctly regardless of the delays of its gates or wires. This is known as the *unbounded delay* model ([11], [16]). A *pure delay* model is also assumed, where the presence of slow inertia is negligible and a pulse of any length can propagate. This is a worst-case model that provides only necessary conditions for the existence of hazards under the *inertial delay* model, where glitches may be suppressed.

The rest of the paper is organized as follows. Section 2 presents basic definitions and an analysis on the necessary conditions for hazard creation and propagation at a circuit line. The proposed hazard-identification function formulation is presented in Section 3. Experimental results are presented in Section 4, and Section 5 concludes.

2 Preliminaries

We first present a brief introduction to combinational hazards and the basic notation used throughout this paper. Next, we discuss the concept of hazard creation (the first instance that a hazard appears at a line), which we then expand to derive the necessary conditions for static and dynamic hazard occurrence at a line.

Combinational hazards are classified as either static or dynamic. A static hazard is the possible occurrence of one or more transient pulses on a signal line whose static value does not change after an input change. A *static-1* hazard is one where the value at the signal line is supposed to remain constant at logic 1 both before and after the input change, but in-between one or more negative pulses ($1 \rightarrow 0 \rightarrow 1$) are generated. *Static-0* hazards are defined analogously, where the value at the signal line is supposed to remain constant at logic 0.

Table 1: Signal values and corresponding notation

Single Value	Symbol
Stable at 1 $\star$	$s1$
Stable at 0 $\star$	$s0$
Rising $\star$	R
Falling $\star$	F
Static-1 †	$s1h$
Static-0 †	$s0h$
Dynamic-R †	dR
Dynamic-F †	dF

$\star$ No hazard † hazard

Sets of Values	Symbol
Initial-1	$1- \in \{s1, s1h, F, dF\}$
Initial-0	$0- \in \{s0, s0h, R, dR\}$
Final-1	$-1 \in \{s1, s1h, R, dR\}$
Final-0	$-0 \in \{s0, s0h, F, dF\}$

A dynamic hazard is the possible occurrence of one or more transient pulses at a signal line whose static value changes after an input change. A dynamic rising (*dynamic-R*) hazard is characterized by one or more positive pulses ($0 \rightarrow 1 \rightarrow 0$) while the value of the signal line is changing from $0 \rightarrow 1$ (rising). Similarly, in a dynamic falling (*dynamic-F*) hazard negative pulses may occur while the signal value changes from $1 \rightarrow 0$ (falling).

Columns 1-2 of Table 1 list the notation that we use to define single signal values. The first four values ($s1, s0, R, F$) represent hazard-free values, and the next four ($s1h, s0h, dR, dF$) represent the four possible combinational hazards. Columns 3-4 of Table 1 show the sets of single values we consider, and their corresponding notation. Each of these sets specify only initial or final signal value requirements that may or may not have hazards, since each one represents a subset of the eight single values.

It is assumed that the changes in the primary input signals of the circuit may or may not take place simultaneously. However, each primary input may change only once during a single transition.

We use the following definitions when discussing the concept of hazard creation.

Definition 2.1. A *primary static* hazard is a static hazard at the output line of a gate whose input lines are all hazard-free.

Definition 2.2. A *primary dynamic* hazard is a dynamic hazard at the output line of a gate whose input lines are all free of dynamic hazards.

The necessary gate input conditions for the creation of primary static hazards at the output of the gate are shown in Fig. 1(a). For an AND or a NOR gate, a primary $s0h$ is created if all three following conditions exist: (i) at least one of its inputs has a hazard-free rising transition (R), (ii)

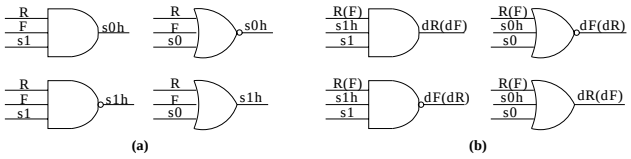

Figure 1: (a) Primary static hazards (b) Primary dynamic hazards

at least one of its inputs has a hazard-free falling transition (F), and (iii) all remaining inputs are hazard-free and stable at a non-controlling value ($s1$ for AND and $s0$ for NOR). A primary $s1h$ is created in a similar manner at the output line of a NAND or an OR gate.

The existence of static hazards is a prerequisite for the creation of primary dynamic hazards. Fig. 1(b) shows the conditions for the occurrence of primary dR and primary dF hazards. For example, a $dR(dF)$ hazard may be created at the output of an AND gate if all of the following conditions occur: (i) at least one of its inputs has a $R(F)$ value, (ii) at least one of its inputs has a $s1h$ value, and (iii) all remaining inputs are hazard-free and stable at a non-controlling value ($s1$).

We expand the conditions for primary hazard creation to describe the generation of a hazard at some arbitrary circuit line. The necessary conditions on the input signals of an AND gate in order for each of the four types of hazards to occur at the gate's output, are given below. The conditions for other types of gates (such as NAND, OR, and NOR) can be derived in an analogous manner.

Static-1 hazards in AND gates (illustrated in Fig. 2):
(i) At least one input has a static-1 hazard ($s1h$)
(ii) All inputs have an initial-1 and final-1 value ($1- \cap -1 \in \{s1, s1h\}$)

Static-0 hazards in AND gates (illustrated in Fig. 3):
Case A:
(i) At least one input has a static-0 hazard ($s0h$)
(ii) Remaining inputs have an initial-1 or final-1 value ($1- \cup -1 \in \{s1, s1h, R, F, dR, dF\}$)
Case B:
(i) At least two inputs have opposite initial and final values
(ii) Remaining inputs have an initial-1 or final-1 value ($1- \cup -1 \in \{s1, s1h, R, F, dR, dF\}$)

Dynamic-R hazards in AND gates (illustrated in Fig. 4):
Case A:
(i) At least one input has a dynamic-R hazard (dR)
(ii) All inputs have a final-1 value (-1)
Case B:
(i) At least one input has a static-1 hazard ($s1h$)
(ii) At least one input has an initial-0 value ($0-$)
(iii) All inputs have a final-1 value (-1)

Dynamic-F hazards in AND gates (illustrated in Fig. 5):
Case A:
(i) At least one input has a dynamic-F hazard (dF)
(ii) All inputs have an initial-1 value ($1-$)
Case B:
(i) At least one input has a static-1 hazard ($s1h$)
(ii) At least one input has a final-0 value (-0)
(iii) All inputs have an initial-1 value ($1-$)

3 Hazard-Identification Functions

We show how the hazard occurrence conditions derived in Section 2 can be used to formulate Boolean functions whose onset is all possible input patterns that may cause the presence of a hazard at a line. A different function is defined for each of the four possible types of hazards.

Let $I = \{i_1, i_2, ..., i_n\}$ be the set of the primary inputs and $L = \{i_1, i_2, ..., i_n, i_{n+1}, ..., i_z\}$ the set of all circuit lines. We define $2 \cdot |I|$ boolean variables , such that for every input $i_k \in I$, variables i_k^1 and i_k^2 represent the initial and final value of input i_k, respectively. Variable set $I^1 = \{i_1^1, i_2^1, ..., i_n^1\}$ denotes the initial primary input values and variable set $I^2 = \{i_1^2, i_2^2, ..., i_n^2\}$ denotes the final primary input values. Functions $f_l^1(I^1)$ and $f_l^2(I^2)$, denote the functionality of a line $l \in L$ with respect to the initial and final input variables, respectively.

Assume an AND gate g with output line l and set of fanin lines denoted by $FI(g)$. We derive the hazard identification function for each of the four types of hazards that may occur at the output of an AND gate. The corresponding functions for other types of gates can be derived based on similar arguments.

The static-1 hazard identification function for line l, denoted by $S1H_l(I^1 \cup I^2)$, is the Boolean function whose onset is all pairs of input patterns that may cause a static-1 hazard to appear at line l. This function is given below:

$$S1H_l = \left[\prod_{\forall i \in FI(g)} (f_i^1 \cdot f_i^2) \right] \cdot \left[\sum_{\forall i \in FI(g)} S1H_i \right] \quad (1)$$

The product term in Equation (1) satisfies condition (ii) which requires every fanin $i \in FI(g)$ of gate g to have both an initial-1 value (f_i^1) and a final-1 value (f_i^2). The summation term in Equation 1 guarantees that at least one of the lines in $FI(g)$ has a static-1 hazard (condition (i)).

The static-0 hazard identification function, denoted by $S0H_l$, is formulated in a similar manner by observing the necessary hazard occurrence conditions derived in Section 2.

For *Case A*, $S0H_l^A(I^1 \cup I^2)$ is given by:

$$S0H_l^A = \sum_{I \subseteq FI(g)} \left[S0H_I \cdot \prod_{J \subset FI(g)} (f_J^1 + f_J^2) \right], \quad (2)$$

where $J \in FI(g) \backslash I$

The outer-most summation is over all possible subsets among the lines in $FI(g)$ (condition (i)). For example, if $FI(g) = \{a, b\}$, then $\sum_{I \subseteq FI(g)} f = f_a + f_b + f_a \cdot f_b$, for

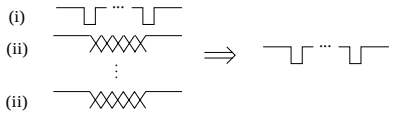

Figure 2: Criteria for $s0h$ to appear at the output of AND gate

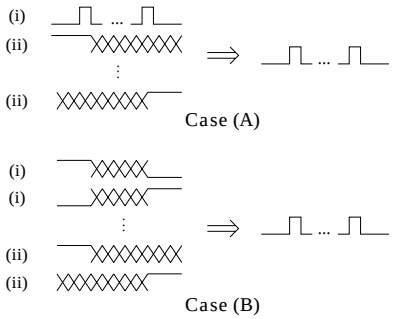

Figure 3: Criteria for $s1h$ to appear at the output of AND gate

some arbitrary function f. The inner product term sets all the remaining fanins $J \in FI(g) \backslash I$ to either an initial-1 or final-1 value (condition (ii)).

Even though the above formulation may seem as computationally expensive, in practice, it is very efficient and this is verified from our experimental results (see Section 4). There are two reasons for this. First, the number of gates' fanins is typically a very small constant in order to avoid excessive gate delays. This is actually a very stringent constraint in most of today's synthesis tools for delay optimization. Second, a considerable number of the terms in the summation of Equation 2 may not have to be computed. When a single fanin i is shown to be static-0 hazard free ($S0H_i = 0$) all the terms of the summation that correspond to subsets that contain i need not be generated. Clearly, the same applies to to pairs of fanins that can not assume a static-0 hazard at the same time, triples of fanins, and so on.

For *Case B*, $S0H_l^B(I^1 \cup I^2)$ is derived by:

$$S0H_l^B = \sum_{I \subset FI(g)} \Big\{ (DF_I + f_I^1 \cdot \overline{f_I^2}) \cdot \prod_{J \subset FI(g)} \Big[(DR_J + \overline{f_J^1} \cdot f_J^2) \cdot \prod_{K \subset FI(g)} (f_K^1 + f_K^2) \Big] \Big\}, \quad (3)$$

where $J \in FI(g) \backslash I, \;\; K \in FI(g) \backslash (I \cup J)$

Function $S0H_l^B$ satisfies the conditions under which *at least* two fanins have opposite initial and final values (condition (i)), and the remaining fanins have either an initial-1 or final-1 value (condition (ii), given by the inner-most product term at the third line in Equation 3).

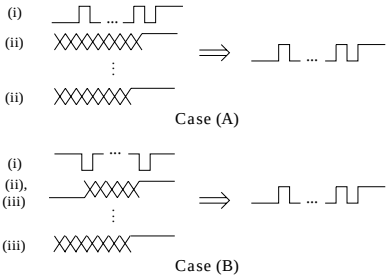

Figure 4: Criteria for dR to appear at the output of AND gate

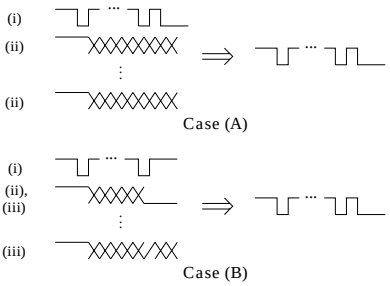

Figure 5: Criteria for dF to appear at the output of AND gate

From Equations 2 and 3, function $S0H_l$ is clearly derived by:

$$S0H_l = S0H_l^A + S0H_l^B \quad (4)$$

The functions for dynamic-R and dynamic-F hazards, denoted by $DR_l(I^1 \cup I^2)$ and $DF_l(I^1 \cup I^2)$ respectively, are given below:

$$DR_l = \left[\prod_{\forall i \in FI(g)} f_i^2 \right] \cdot \left\{ \sum_{I \subseteq FI(g)} DR_I + \sum_{I \subset FI(g)} \left[S1H_I \cdot \sum_{J \subset FI(g)} \overline{f_J^1} \right] \right\}, \quad (5)$$

where $J \in FI(g) \backslash I, \;\; K \in FI(g) \backslash (I \cup J)$

$$DF_l = \left[\prod_{\forall i \in FI(g)} f_i^1 \right] \cdot \left\{ \sum_{I \subseteq FI(g)} DF_I + \sum_{I \subset FI(g)} \left[S1H_I \cdot \sum_{J \subset FI(g)} \overline{f_J^2} \right] \right\}, \quad (6)$$

where $J \in FI(g) \backslash I, \;\; K \in FI(g) \backslash (I \cup J)$

The first product in Equation 5 (resp. 6) satisfies the condition that all inputs must have a final-1 (resp. initial-1) value. The expression in the curling brackets satisfies condition (i) of *Case A* or conditions (i) and (ii) of *Case B*, for both functions.

The termination conditions for the recursive definitions of the hazard identification functions are clearly $S1H_l = S0H_l = DR_l = DF_l = 0, \; l \in I$, since all primary inputs are assumed to be hazard-free. We generate the hazard identification functions for all circuit lines by a single topological traversal of the circuit, starting at the primary input lines.

We observe that there is some redundancy in the proposed function formulations. However, this redundancy is useful for computational efficiency. For example, in the definition of $S1H_l$ (Equation 1), if $\prod_{\forall i \in FI(g)}(f_i^1 \cdot f_i^2)$ is not satisfied then a $s1h$ value can never occur at the output line of gate g because it is not possible to set all of the fanins of g to value 1 simultaneously. This case can be identified without spending any time computing the summation in Equation 1, which is the more expensive of the two parts in terms of computational complexity. The same applies for $\prod_{\forall i \in FI(g)} f_i^2$ in Equation 5 and $\prod_{\forall i \in FI(g)} f_i^1$ in Equation 6.

Finally, the set of *all* hazardous input transitions at output line l of a gate g, denoted by $\mathcal{H}_l(I^1 \cup I^2)$, is given by:

$$\mathcal{H}_l = S1H_l + S0H_l + DR_l + DF_l \tag{7}$$

4 Experimental Results

We developed the proposed approach using BDDs [5] to represent the boolean functions. The implementation was done in C language, on top of the BDD package of [18], and run on a 900MHz SunBlade 1000 workstation. Experimental results for the ISCAS'85 combinational circuits and the full-scanned versions of the ISCAS'89 circuits are presented.

Table 2 lists our findings when function $\mathcal{H}_l$ (from Equation 7) was generated per circuit line. Column 1 shows the circuit name and Columns 2-3 and 4-5 list the results obtained when the ordering of the BDD variables is randomly fixed and when automatic reordering is allowed, respectively. Columns 2 and 4 show the CPU time (in seconds) and Columns 3 and 5 show the memory utilization (in MBs). The best time is shown is bold numbers in either Column 2 or Column 4. We have observed that for the cases that the approach fails when the BDD variables' order remains fixed (indicated by † in Columns 2-3), the majority of the CPU time when dynamic reordering is enabled (Column 4) is spend on variable reordering.

It is well known that certain functions cannot be represented efficiently by BDDs under any variable ordering. An example of such functions is circuit c6288 which is a 16-by-16 multiplier (indicated by $\star$ in Table 2, Column 1). In such cases, we can generate partial solutions by fixing a small cardinality subset $\mathcal{I}'$ of the input variable set I to either 0 or 1. Iteration of this approach over all the possible $2^{\mathcal{I}'}$ combinations guarantees a complete solution. The results given for circuit c6288 in Columns 4-5 were derived based on this simplification heuristic, when 8 variables were fixed over 2^8 iterations. The time reported for this circuit is in hours.

Columns 6 and 7 of Table 2 list the percentage of CPU time and memory reduction obtained from the proposed approach over the glitchy signal transition set generation procedure of [10]. We implemented the latter procedure using also BDDs, and run it under both fixed variable ordering and dynamic reordering. The results reported are the best ones obtained under any of the two ordering methods. We observe that in several cases the reduction is considerable. Moreover, our approach succeeded in several cases where the approach in [10] failed due to excessive space requirements (these cases are indicated by ‡ in Columns 6-7).

We also used the simplification (variable fixing) method as a heuristic for examining the decision problem of hazard existence at a circuit line. Instead of considering all possible $2^{\mathcal{I}'}$ iterations, we now terminate as soon as the line hazard function under some iteration is not the constant 0 function. In practice, if a line has a hazard it will be created by several input conditions, and, thus, the variable fixing method quickly determines if a hazard exists at the line. If a line is hazard-free, all iterations of the variable fixing method must be carried through before determining that a hazard does not exits. However, each iteration terminates relatively much faster than the overall approach (where no variables are fixed), and, as long as the cardinality of $\mathcal{I}'$ is kept small, the variable fixing approach is very efficient. The results are shown in Columns 8-9 of Table 2. The size of $\mathcal{I}'$ was set between 5-8 inputs. These results demonstrate that the proposed approach is very promising and its performance is only expected to be improved when non-canonical representations, along with sophisticated algorithms for determining satisfiability are employed.

Table 2: Results for the ISCAS'89 and ISCAS'85 circuits.

Circuit	FIXED ORDERING		DYNAMIC REORDERING		REDUCTION from [10]		DECISION PROBLEM	
	Time (secs)	Mem. (MBs)	Time (secs)	Mem. (MBs)	Time (%)	Mem. (%)	Time (secs)	Mem. (MBs)
s298	**0.05**	1.3	0.06	1.3	70.5	65.0	0.02	0.7
s344	**0.13**	1.5	**0.13**	1.5	59.4	21.4	0.06	0.9
s349	**0.11**	1.5	**0.11**	1.5	60.7	27.1	0.06	0.8
s382	0.08	1.3	**0.06**	1.3	81.2	61.1	0.01	0.6
s386	**0.31**	4.1	**0.31**	4.1	22.5	9.0	0.20	1.5
s400	**0.07**	1.3	0.08	1.3	75.0	53.6	0.06	0.9
s420	†	†	**11.93**	11.6	‡	‡	2.11	7.3
s444	0.15	2.2	**0.14**	2.2	56.2	20.2	0.09	1.0
s526	0.14	2.1	**0.13**	2.1	45.8	22.3	0.09	1.0
s526n	**0.13**	2.1	0.14	2.1	44.9	22.7	0.08	1.0
s641	**5.09**	35.4	30.84	34.0	67.5	3.1	10.13	20.9
713	**5.44**	37.0	30.17	34.8	58.5	3.3	11.09	21.5
s820	68.42	79.8	**11.01**	29.1	-61.1	-8.9	4.09	10.2
s832	69.86	80.2	**12.63**	29.1	-65.3	-5.1	4.61	9.9
s953	**1.79**	16.3	1.82	16.3	35.4	2.1	0.72	10.3
s1196	**5.20**	35.4	17.44	28.3	27.0	1.1	10.31	13.5
s1238	**5.18**	35.2	17.44	28.2	24.5	1.5	9.22	13.3
s1423	†	†	**146.75**	18.6	‡	‡	11.11	11.9
s1488	**3.52**	30.8	3.57	30.8	33.3	15.0	0.91	10.3
s1494	**3.44**	30.7	3.44	30.7	29.8	13.5	0.89	10.1
s9234	†	†	**1427.48**	204.9	‡	‡	13.01	33.3
c880	**101.61**	101.2	1799.62	60.3	17.1	2.7	8.4	32.0
c1355	†	†	**59.7h**	935.1	‡	‡	210.1	65.3
c1908	**3009.17**	133.1	5190.17	95.7	12.0	1.3	56.3	51.3
c2670	**31.14**	54.5	433.19	33.2	25.3	12.2	1.5	15.9
c3540	†	†	**1915.83**	60.5	43.4	38.4	20.3	41.2
c5315	†	†	**1650.72**	49.4	52.5	45.3	13.9	28.4
c6288$\star$	†	†	**63.36h**	950.0	‡	‡	891.5	73.3
c7552	**26.99**	58.0	1285.43	57.6	22.2	11.2	6.4	53.3

† Approach failed due to space requirements exceeding 1,000 MBs
‡ [10] failed to terminate using up to 1,000 MBs

5 Conclusion

We have presented a systematic methodology for generating functional descriptions of the complete set of hazardous input transitions in combinational circuits. The proposed function formulation is based on the identification of all the necessary conditions at a gate's input signals, for a specific type of hazard to appear at the gate's output. The experimental results obtained demonstrate that the proposed approach is very promising. The approach also supports identification of hazards caused by single-input changes (SIC), by restricting (cofactoring) the generated functions appropriately. This is of particular importance in the context of synthesis of hazard-free circuits since it is well known that hazards which are caused by SICs can always be eliminated.

References

[1] D. Bhattacharya, P. Agrawal and V. D. Agrawal, "Test Pattern Generation for Path Delay Faults using Binary Decision Diagrams", *IEEE Trans. on Computers*, Vol. 44, No. 3, pp. 434-447, March 1995.

[2] J. Beister, "A unified approach to combinational hazards", *IEEE Trans. on Computers*, Vol. C-23, No. 6, pp. 566-575, June 1974.

[3] J. G. Bredeson and P. T. Hulina, "Elimination of static and dynamic hazards for multiple input changes in combinational switching circuits", *Information and Control*, Vol. 20, pp. 114-124, 1972.

[4] M. A. Breuer and R. L. Harrison, "Procedures for eliminating static and dynamic hazards in test generation", *IEEE Trans. on Computers*, Vol. C-23, No. 10, pp. 1069-1077, October 1974.

[5] R. Bryant, "Graph-based algorithms for boolean function manipulation". *IEEE Trans. on Computers*, Vol. C-35, No. 8, pp. 677-691, August 1986.

[6] J. Brzozowski and Z. Esik, "Hazard algebras", *Maveric Research Report 00-2*, Department of Computer Science, University of Waterloo, December 2001.

[7] E. B. Eichelberger, "Hazard detection in combinational and sequential switching circuits", *IBM Journal*, Vol. 9, No. 2. pp. 90-99, March 1965.

[8] D. A. Huffman, "The design and use of hazard-free switching networks", *Journal of ACM*, Vol. 4, pp. 47-62, 1957.

[9] W. Ke and P. Menon, "Synthesis of Delay-Verifiable Combinational Circuits", *IEEE Trans. on Computers*, Vol. 44, No. 2, pp. 213-222, February 1995.

[10] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, "Digital Sensitivity: Predicting Signal Interaction using Functional Analysis", *Proc. of ICCAD*, pp. 536-541, November 1996.

[11] B. Lin and S. Devadas, "Synthesis of hazard-free multilevel logic under multiple-input changes from binary decision diagrams", *IEEE Trans. on CAD*, Vol. 14, No. 8, pp. 974-985, August 1995.

[12] C. J. Lin and S. M. Reddy, "On delay fault testing in logic circuits", *IEEE Trans. on CAD*, Vol. CAD-6, No. 5, pp. 694-703, September 1987.

[13] E. J. McCluskey, "Transient behavior of combinational logic networks", *Redundancy Techniques for Computing Systems*, Spartan Books, pp. 9-46, 1962.

[14] P. C. McGeer, A. Saldanha, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Timing analysis and delay-fault test generation using path-recursive functions", *Proc. of ICCAD*, pp. 180-183, 1991.

[15] M. Michael, "Test-based timing verification using functional techniques", PhD Dissertation, ECE Dept., Southern Illinois University at Carbondale, July 2002.

[16] S. M. Nowick and D. L. Dill, "Exact two-level minimization of hazard-free logic with multiple-input changes", *IEEE Trans. on CAD*, Vol. 14, No. 8, pp. 986-997, August 1995.

[17] M. Sivaraman and A. J. Strojwas, "Primitive Path Delay Faults: Identification and Their Use in Timing Analysis", *IEEE Trans. on CAD*, Vol. 19, No. 11, pp. 1347-1362, November, 2000.

[18] F. Somenzi, "CUDD: CU Decision Diagram Package", ECE Dept., The University of Colorado at Boulder, release 2.3.0, 1999.

[19] J. Sosnowski, "An algebraic approach towards transient analysis of asynchronous circuits", *Digital Processes*, Vol. 4, No. 1, pp. 45-66, 1978.

[20] A. Thayse, "Transient analysis of logical networks applied to hazard detection", *Philips Research Report*, Vol. 25, pp. 261-336, 1970.

[21] A. Thayse and M. Davio, "Boolean differential calculus and its application to switching theory", *IEEE Trans. on Computers*, Vol. C-22, pp. 409-419, April 1973.

[22] S. H. Unger, *Asynchronous sequential switching circuits*, New York:Wiley-Interscience, 1969, (Reissued by Krieger, Malabar Fla., 1983).

[23] S. H. Unger, "Hazards, critical races, and metastability", *IEEE Trans. on Computers*, Vol. 46, No. 6, pp. 754-768, June 1995.