Journal of Universal Computer Science, vol. 5, no. 11 (1999), 765-776
submitted: 31/5/99, accepted: 26/11/99, appeared: 28/11/99 © Springer Pub. Co.

Matrix Method to Detect Logic Hazards in Combinational Circuits with EX-OR Gates

E. C. Tan
(Nanyang Technological University, Singapore
asectan@ntu.edu.sg)

M. H. Ho
(Nanyang Technological University, Singapore)

Abstract: A matrix method is extended to include the detection of logic hazards in combinational logic circuits involving EX-OR gates. Essentially, the method generates 0- and 1-sets, or *P*- and *S*-sets, of all nodes in each gate level of a circuit progressively until it reaches the output of the circuit. The sets generated are subsequently used to determine the existence of static or dynamic hazards.

Key Words: Hardware - Logic Hazards - Exclusive-OR Gates - Matrix Method

1 Introduction

Hazards are unwanted switching transients which may appear at the output of a combinational circuit and are caused by propagation delays of the input signals. There are two types of hazards, viz., function and logic hazards, if we classify them by specific causes. A function hazard is associated with multiple-input changes while a logic hazard is caused by a single- input change. This paper will only deal with logic hazards. We can further classify hazards as being static or dynamic by their output waveforms. The presence of undesirable glitches at the output during the transition of two input states that have the same steady-state output is called a static hazard. It is a static 0 (1)-hazard if the steady-state output is 0 (1). Dynamic hazard occurs when the steady-state output values of two input states are different and the output experiences more than one change.

The study of hazards in combinational logic circuits in the past [Beister 1974, McCluskey 1986, Tinder 1991, Almaini 1994, Wakerly 1994, Nelson et al. 1995] has mainly focused on logic elements other than exclusive-OR (EX-OR) gates. Comparing to the traditional AND/OR formulation of Boolean functions, the alternative AND/EX-OR logic approach based on Reed-Muller (RM) expansions [Green 1986, Almaini 1994] has advantages in certain types of realisations. For examples, RM circuits have desirable features of ease of complementing and testing [Reddy 1972], and AND/EX-OR PLAs often require fewer product terms than corresponding AND/OR PLAs [Sasao and Besslich 1990]. In addition, logic functions that do not minimise well in the Boolean domain can be reduced substantially if implemented in RM logic [Green 1986]. Although the Karnaugh map can be used to detect hazards for all types of digital gates, it is limited to functions with no more than six variables. The techniques of exhaustive testing (an extension of the graphical map method) and compatibility evaluation of the product terms of a modulo-2 sum have been proposed [Ouyang and Tran 1994] for hazard detection of EX-OR gates. However, the procedures are unwieldy.

A matrix method for detecting hazards in logic circuits has been reported [Heal and Page 1993]. However, it only describes the generations of 0- and 1-sets [McCluskey 1986] for detecting static hazards in combinational circuits with AND/OR/NAND/NOR gates. This paper has two purposes. It will show how to adopt and modify the matrix method to accommodate EX-OR gates also in combinational circuits, and to incorporate the generation of *P*- and *S*-sets [McCluskey 1986] for detecting dynamic hazards. The advantage of the matrix method is that it starts with the circuit inputs and then follows signals through the circuit one element level at a time, inspecting each element output along the way to locate hazards.

2 Set Generation

Static logic hazards are caused by two signals (such as x and x') with opposite steady-state values taking on the same value when the network signals are changing [McCluskey 1986]. In analyzing such hazards, literals such as such as x and x' must be taken as two different variables rather than complements of the same variable. The relations $xx' = 0$, $x + x' = 1$ and other theorems based on these two relations must not be used to rewrite expressions. The procedure to analyze static logic hazards starts by forming either a sum-of-products (SOP) or a product-of-sums (POS) expression for the network function. The Boolean relations $x + xy = x$ and $x(x + y) = x$ are used to eliminate redundant product terms or sum factors. Each product term in a SOP expression formed by this procedure corresponds to a "1-set" of the network. For example, if the SOP expression is $f = a'ab + abb' + abc + b'd$, the corresponding 1-sets are $\{a', a, b\}$, $\{a, b, b'\}$, $\{a, b, c\}$ and $\{b', d\}$. Similarly, each sum factor in a POS expression obtained by this procedure corresponds to a "0-set" of the network. For example, if the POS expression is $f = (a + b')(a + d)(b + b')(b + d)(c + a' + b')$, the corresponding 0-sets are $\{a, b'\}$, $\{a, d\}$, $\{b, b'\}$, $\{b, d\}$ and $\{c, a', b'\}$.

In terms of gate behaviour, a set of variables (with or without complement symbols) is a 0 (1)-set of an element output if (i) whenever all of the variables are equal to 0 (1), the element output is equal to 0 (1); and (ii) if any variable is removed, condition (i) no longer holds [McCluskey 1986]. The existence of a static logic hazard at an element output can be determined by analyzing the 0-sets and 1-sets of the output signal. To facilitate the generation of these sets at input nodes, the matrix method [Heal and Page 1993] requires that any AND and NAND gates in a given combinational circuit are first converted to augmented-OR gates, where an augmented-OR gate [Nelson 1995] is an OR gate with inverters incorporated into one or more of its inputs (see Figs. 1 and 2). Any OR gates in the circuit remain unchanged, and the output bubble of a NOR gate can be cancelled out at any input where a bubble already exists.

Rules for 0-set and 1-set generation using an augmented-OR gate can be formulated [Heal and Page 1993] from Fig.1. The output $f = 0$ if $i_1 = \ldots = i_m = 0$ and $c_1 = \ldots = c_n = 1$. The condition for setting i (c) input equal to 0 (1) is defined by its 0 (1)-sets. For example, if the 0-sets for i_1 are $\{U\}$ and $\{V, W\}$, then i_1 can be set to 0 either by making $U = 0$ or by making $V = W = 0$; if the 1-sets for c_1 are $\{X\}$ and $\{Y, Z\}$ then c_1 can be set to 1 either by making $X = 1$ or by making $Y = Z = 1$. A 0-set or 1-set is said to be unstable if it contains at least one pair of complementary variables. A 1(0)-set is said to be active if all its variables are equal to 1(0).

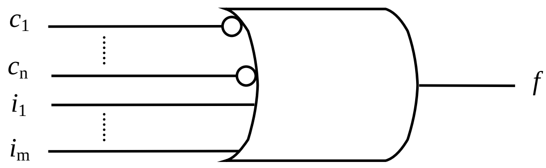


Figure 1. Generalized augmented-OR gate

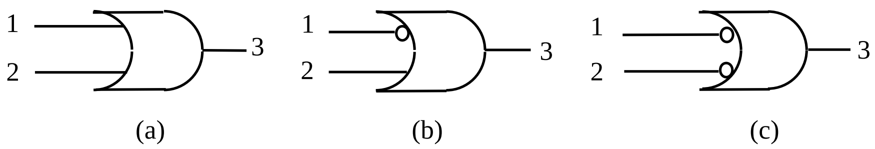


Figure 2. (a) OR gate (b) Augmented-OR gate (1 inverted input)
(c) Augmented-OR gate (2 inverted inputs)

Let $(\text{0-set})_m$ and $(\text{1-set})_m$ be the 0-set and 1-set respectively at node m (where m is an integer), '$\times$' the Cartesian product, and '$\cup$' the set union. The above set-generation rules can be more compactly defined as follows. For Fig. 2(a),

$$(\text{0-set})_3 = (\text{0-set})_1 \times (\text{0-set})_2 \quad (1)$$
$$(\text{1-set})_3 = (\text{1-set})_1 \cup (\text{1-set})_2 \quad (2)$$

For Fig. 2(b),

$$(\text{0-set})_3 = [(\text{1-set})_1]' \times [(\text{1-set})_2]' \quad (3)$$
$$(\text{1-set})_3 = [(\text{0-set})_1]' \cup (\text{1-set})_2 \quad (4)$$

For Fig. 2(c),

$$(\text{0-set})_3 = [(\text{1-set})_1]' \times [(\text{1-set})_2]' \quad (5)$$
$$(\text{1-set})_3 = [(\text{0-set})_1]' \cup [(\text{1-set})_2]' \quad (6)$$

3 EX-OR Gate

Let us consider an EX-OR gate (Fig. 3(a)). If we use the Boolean relation, $y = x_1'x_2 + x_1x_2'$, to convert it into the augmented-OR form as in Fig. 3(b), we have $(\text{0-set})_7 = (\{x_1, x_1'\}\{x_1, x_2\}\{x_1', x_2'\}\{x_2, x_2'\})$ and $(\text{1-set})_7 = (\{x_1', x_2\}\{x_1, x_2\})$. From the 0-set, we have two unstable sets, $\{x_1, x_1'\}$ and $\{x_2, x_2'\}$. Since the EX-OR gate is built as a monolithic chip, the two unstable sets should not be there. Thus in a network, we cannot replace an EX-OR gate with its augmented-OR form as shown in Fig. 3(b). However, we can treat an EX-OR gate as an autonomous gate and define the set generation rule of the EX-OR gate in Fig. 3(a) as

$$(\text{0-set})_3 = \{[(\text{1-set})_1]' \times [(\text{1-set})_2]'\} \cup \{(\text{0-set})_1 \times (\text{0-set})_2\} \quad (7)$$
$$(\text{1-set})_3 = \{[(\text{0-set})_1]' \times (\text{1-set})_2\} \cup \{(\text{1-set})_1 \times [(\text{0-set})_2]'\} \quad (8)$$

If the input of an EX-OR gate has an inverter, simply replace the 0-set at that node by the complement of its 1-set, and vice versa.

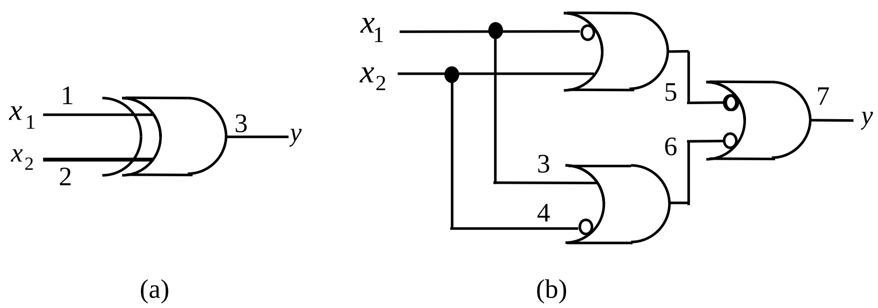


Figure 3. (a) EX-OR gate (b) Its augmented-OR form

4 Static Hazards

To determine both the static logic 1-hazards and 0-hazards from the network 1-sets or 0-sets, we use the following theorem from [McCluskey 1986]:

Theorem: A static logic 1(0)-hazard exists at an element output if and only if (a) There is a pair of input states that both produce a 1(0)-state at the element output, and (b) There is no stable 1(0)-set that is active for both input states.

4.1 Algorithm

Let N = the total number of nodes in the circuit, L = last gate level, and n_k = number of nodes at the kth gate level. The algorithm to detect static hazards in a combinational logic circuit is given as follows [Heal and Page 1993]:

Step 1: Convert any AND and NAND gates to augmented-OR gates.

Step 2: Label the nodes.

Step 3: Construct the connection matrix, **C** (N x N).

Step 4: At gate level k = 0, create the initial set matrix (N entries), $\mathbf{S}_0 = [s_1 \;\; s_2 \; \ldots \; s_{n0} \;\; 0 \;\; 0 \; \ldots \; 0]^T$, where s_j consists of both the 0- and 1-sets at the jth node. Based on the generated 0- and 1-sets, examine static hazards accordingly.

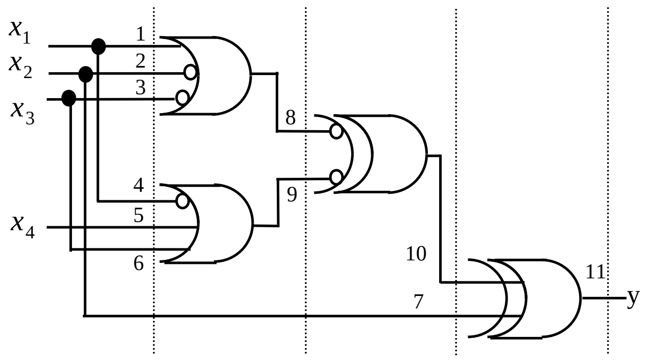
Step 5: To advance to the next gate level to obtain the next set information, perform the matrix operation, $\mathbf{S}_1 = \mathbf{S}_0 + \mathbf{C} * \mathbf{S}_0$ so that $\mathbf{S}_1 = [s_1 \;\; s_2 \; \ldots \; s_{n0} \;\; s_{n0+1} \; \ldots \; s_{n0+n1} \;\; 0 \;\; 0 \; \ldots \; 0]^T$

Step 6: Examine the 0- and 1-sets of s_{n0+1}, s_{n0+2}, ..., s_{n0+n1} for static hazards.

Step 7: Repeat steps 5 and 6 until $k = L$ and $\mathbf{S}_L$ has nonzero entries, i.e., $\mathbf{S}_L = [s_1 \;\; s_2 \; ... \; s_N]^T$.

In general, $\mathbf{S}_{p+1} = \mathbf{S}_p + \mathbf{C} * \mathbf{S}_p$, where p = 0, 1, ..., L - 1, and $k = p + 1$. In the original algorithm by Heal and Page (1993), an N x N 'sets' matrix $\mathbf{S}_p$ is used even though only the diagonal elements of $\mathbf{S}_p$ are involved in the matrix multiplication. Furthermore, although the results of multiplications are stored in non-diagonal rows of $\mathbf{S}_p$, they are there only temporarily. Thus, in the above algorithm, it is sufficient to use $\mathbf{S}_p$ with a dimension of N x 1 only. For filling the connection matrix **C**, every entry in a row represents the presence of an input. An 'i' entry represents an input signal that is not inverted while a 'c' entry means that the input signal is inverted. The operator $*$ in the algorithm is defined as follows: $0 * 0 = 0$, $0 * s_i = 0$, $c * 0 = 0$, $i * 0 = 0$, $c * s_i = s_i$ and $i * s_i = s_i$.

4.2 Example



	$\mathbf{S_0}$	==>	$\mathbf{S_1}$	==>	$\mathbf{S_2}$	==>	$\mathbf{S_3}$
k =	0		1		2		3
n_k =	6		2		2		1

Figure 4. Circuit for illustration

Fig. 4 shows a circuit in which any AND or NAND gates have been converted to augmented-OR gates, and all nodes have been labeled. The connection matrix **C** is given below. We obtain $\mathbf{S}_0 = [s_1 \ldots s_7\ \ 0 \ldots 0]^T$ in which each node-set, s_j, j = 1..7, consists of its own input variable in the 0- and 1-sets. Furthermore, $\mathbf{S}_1 = [s_1 \ldots s_9\ \ 0 \ldots 0]^T$ and $\mathbf{S}_2 = [s_1 \ldots s_{10}\ \ 0 \ldots 0]^T$ where s_8, s_9 and s_{10} are given in Table 1. No static hazard is detected at this stage. Lastly, $\mathbf{S}_3 = [s_1 \ldots s_{11}]^T$.

At node 11, we have

0-set = $(\{x_1, x_2\}\{x_2, x_3'\}\{x_2, x_4'\}\{x_1, x_1', x_2\}\{x_1, x_2, x_3\}\{x_1, x_2', x_3'\}\{x_1', x_2, x_3'\}\{x_1', x_2, x_4'\}\{x_2, x_3, x_3'\}\{x_2, x_3, x_4'\}\{x_1, x_2', x_3', x_4'\}\{x_1', x_2', x_3, x_4\}\{x_1', x_2, x_2', x_3, x_4\}\{x_1, x_1', x_2, x_2', x_3, x_3', x_4\})$

1-set = $(\{x_1, x_1', x_2\}\{x_1, x_2, x_3\}\{x_1, x_2, x_4\}\{x_1', x_2, x_2'\}\{x_1', x_2, x_3'\}\{x_2, x_2', x_3\}\{x_2, x_2', x_4\}\{x_2, x_3, x_3'\}\{x_2, x_3', x_4\}\{x_1, x_2', x_3', x_4'\}\{x_1', x_2, x_2', x_3\}\{x_1', x_2, x_2', x_3, x_4\}\{x_1, x_1', x_2, x_3, x_3', x_4'\})$.

$$\mathbf{C} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ i&c&c&0&0&0&0&0&0&0&0 \\ 0&0&0&c&i&i&0&0&0&0&0 \\ 0&0&0&0&0&0&0&c&c&0&0 \\ 0&0&0&0&0&0&i&0&0&i&0 \end{bmatrix}$$

Node-set	1-sets	0-sets
s_8	$\{x_1\}\{x_2'\}\{x_3'\}$	$\{x_1, x_2', x_3'\}$
s_9	$\{x_1'\}\{x_3\}\{x_4\}$	$\{x_1', x_3, x_4\}$
s_{10}	$\{x_1, x_3', x_4'\}\{x_1', x_2, x_3\}$ $\{x_1, x_2', x_3', x_4'\}\{x_1', x_2, x_3, x_4\}$	$\{x_1, x_1'\}\{x_1, x_2\}\{x_1, x_3\}\{x_1', x_3'\}$ $\{x_1', x_4'\}\{x_2, x_3'\}\{x_2, x_4'\}\{x_3, x_3'\}$ $\{x_3, x_4'\}\{x_1, x_1', x_2', x_3, x_3', x_4\}$

Table 1. Set information for static hazard.

Recall that in testing a network for static logic hazards, it is only necessary to generate *either* the 1-sets *or* the 0-sets [McCluskey 1986, page 86], although both types of sets are calculated in the example. For many networks, it is generally easier to calculate one type of sets rather than the other type.

Examine only the 1-set. For the unstable set, {x_1', x_2, x_2', x_3, x_4}, there is no way to make any of the stable sets active. Using the theorem for static logic hazard stated earlier, when x_1 = 0, x_3 = 1, x_4 = 1 and x_2 changes, static logic 0-hazard is possible. Furthermore, there is also another static logic 0-hazard that may occur in the circuit when x_1 = 0, x_3 = 1, x_4 = 0, with x_2 changing.

5 Dynamic Hazards

The matrix method can also include dynamic hazard detection. In a dynamic hazard, an output may change several times for a single change in input. As a result, we need to trace all the paths taken by an input-variable and differentiate them by using different subscripts. This is an indirect but important way of associating order of signals as the 'timing' aspects is required to detect dynamic hazards.

All the 1-sets are replaced by *P*-sets and 0-sets by *S*-sets [McCluskey 1986]. In essence, A *P*-set (*S*-set) is defined and computed in the same manner as a 1-set (0-set) with the additional requirement that whenever an input variable can propagate to the output along more than one path, each appearance of that variable must include a label identifying which path corresponds to the instance of that variable. That is, we need to trace all the paths taken by an input-variable and differentiate them by different subscripts. This will be satisfied if whenever a variable is propagated through a labelled lead, the lead label is added as a subscript to that variable. To determine dynamic hazards, we use the following corollary from [McCluskey 1986, page 92]:

Corollary: If a network contains any dynamic hazards, it must contain at least one pair of unstable sets, one of which is a P-set (1-set) and the other of which is a S-set (0-set); the pair of unstable sets must share the same unstable variable, and any other variable that appears in both sets must be complemented in one of the sets and uncomplemented in the other.

Of course, one can also use the theorem in [McCluskey 1986, page 90] to examine the existence of dynamic hazard. However, four conditions in the theorem must be tested in order to detect dynamic hazard.

5.1 Algorithm

The algorithm is the same as in Section 4.1 except modifications in the following three steps:

Step 2: Label the nodes. For each fan-out, a subscript with the same label along the path is added to the variable.

Step 4: Create $\mathbf{S}_k$ (N x 1) and enter the *P*-sets and *S*-sets for the circuit inputs in the appropriate row. All the variables of the primary input will take up the subscript of the input line.

Step 6: For all the rows with new entries, combine the set entries and determine any dynamic hazards. If the new set is a fan-out branch, the subscript of the line will be added to the subscript of all the variables in the new set.

5.2 Example

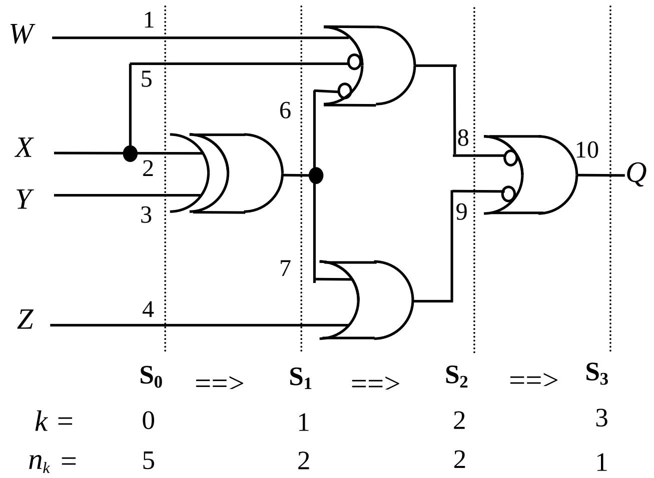


Figure 5. Circuit in converted and labelled form

Consider the circuit in Fig. 5 for determining the existence of any possible dynamic logic hazard.

Step 1: Perform necessary conversions so that the circuit contains only EX-OR and/or augmented-OR gates.

Step 2: Label all the nodes. (For each fan-out, a subscript with the same label along the path will be added to the variable later.)

Step 3: The following connection matrix (N = 10) is obtained:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & i & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & i & i & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 0 & c & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & 0 & 0 & i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & c & 0 \end{bmatrix}$$

Step 4: The first five rows in **C** are all primary inputs. Thus, from Fig. 5, we have $\mathbf{S}_0 = [s_1\ \ s_2\ \ s_3\ \ s_4\ \ s_5\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$ in which the respective *P*-sets and *S*-sets for s_1 , …, s_5 are given in Table 2 below. After checking with the above corollary for all the *P*-sets and *S*-sets generated so far, no dynamic hazard exists.

Node-set	*P*-sets	*S*-sets
s_1	$\{W_1\}$	$\{W_1\}$
s_2	$\{X_2\}$	$\{X_2\}$
s_3	$\{Y_3\}$	$\{Y_3\}$
s_4	$\{Z_4\}$	$\{Z_4\}$
s_5	$\{X_5\}$	$\{X_5\}$

Table 2. Set information for s_1 , …, s_5.

Step 5: $\mathbf{S}_1 = \mathbf{S}_0 + \mathbf{C} * \mathbf{S}_0 = [s_1\ \ s_2\ \ s_3\ \ s_4\ \ s_5\ \ s_6\ \ s_7\ \ 0\ \ 0\ \ 0]^T$ in which the respective *P*-sets and *S*-sets for s_6 and s_7 are given in Table 3 below. Note that because the output of the EX-OR gate in Fig. 5 is a fan-out node, a subscript with the same label along the path is added to the variable according to the method in [McCluskey 1986].

Node-set	*P*-sets	*S*-sets
s_6	$\{X_{2,6}, Y_{3,6}'\}\ \{X_{2,6}', Y_{3,6}\}$	$\{X_{2,6}, Y_{3,6}\}\ \{X_{2,6}', Y_{3,6}'\}$
s_7	$\{X_{2,7}, Y_{3,7}'\}\ \{X_{2,7}', Y_{3,7}\}$	$\{X_{2,7}, Y_{3,7}\}\ \{X_{2,7}', Y_{3,7}'\}$

Table 3. Set information for s_6 and s_7.

Step 6: After checking with the above corollary for all the *P*-sets and *S*-sets generated so far, no dynamic hazard exists.

Step 7: Go to Step 5.

Step 5: $\mathbf{S}_2 = \mathbf{S}_1 + \mathbf{C} * \mathbf{S}_1 = [s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ s_7\ s_8\ s_9\ 0]^T$ in which the respective *P*-sets and *S*-sets for s_8 and s_9 are given in Table 4 below.

Node-set	*P*-sets	*S*-sets
s_8	$\{W_{1,8}\}$ $\{X_{5,8}'\}$ $\{X_{2,6,8}, Y_{3,6,8}\}$ $\{X_{2,6,8}', Y_{3,6,8}'\}$	$\{W_{1,8}, X_{5,8}', X_{2,6,8}, Y_{3,6,8}'\}$ $\{W_{1,8}, X_{5,8}', X_{2,6,8}', Y_{3,6,8}\}$
s_9	$\{X_{2,7,9}, Y_{3,7,9}'\}$ $\{X_{2,7,9}', Y_{3,7,9}\}$ $\{Z_{4,9}\}$	$\{Z_{4,9}, X_{2,7,9}, Y_{3,7,9}\}$ $\{Z_{4,9}, X_{2,7,9}', Y_{3,7,9}'\}$

Table 4. Set information for for s_8 and s_9.

Step 6: After checking with the above corollary for all the *P*-sets and *S*-sets generated so far, no dynamic hazard exists.

Step 7: Go to Step 5.

Step 5: $\mathbf{S}_3 = \mathbf{S}_2 + \mathbf{C} * \mathbf{S}_2 = [s_1\ s_2\ s_3\ s_4\ s_5\ s_6\ s_7\ s_8\ s_9\ s_{10}]^T$ in which the *P*-sets and *S*-sets for s_{10} are given in Table 5 below.

Node-set	*P*-sets	*S*-sets
s_{10}	$\{W_{1,8,10}', X_{5,8,10}, X_{2,6,8,10}, Y_{3,6,8,10}'\}$ $\{W_{1,8,10}', X_{5,8,10}, X_{2,6,8,10}', Y_{3,6,8,10}\}$ $\{Z_{4,9,10}', X_{2,7,9,10}, Y_{3,7,9,10}\}$ $\{Z_{4,9,10}', X_{2,7,9,10}', Y_{3,7,9,10}'\}$	$\{W_{1,8,10}', X_{2,7,9,10}', Y_{3,7,9,10}\}$ $\{X_{5,8,10}, X_{2,7,9,10}', Y_{3,7,9,10}\}$ $\{X_{2,6,8,10}, X_{2,7,9,10}', Y_{3,6,8,10}, Y_{3,7,9,10}\}$ $\{X_{2,6,8,10}', X_{2,7,9,10}', Y_{3,6,8,10}', Y_{3,7,9,10}\}$ $\{W_{1,8,10}', X_{2,7,9,10}, Y_{3,7,9,10}'\}$ $\{X_{5,8,10}, X_{2,7,9,10}, Y_{3,7,9,10}'\}$ $\{X_{2,6,8,10}, X_{2,7,9,10}, Y_{3,6,8,10}, Y_{3,7,9,10}'\}$ $\{X_{2,6,8,10}', X_{2,7,9,10}, Y_{3,6,8,10}', Y_{3,7,9,10}'\}$ $\{W_{1,8,10}', Z_{4,9,10}'\}$ $\{X_{5,8,10}, Z_{4,9,10}'\}$ $\{X_{2,6,8,10}, Y_{3,6,8,10}, Z_{4,9,10}'\}$ $\{X_{2,6,8,10}', Y_{3,6,8,10}', Z_{4,9,10}'\}$

Table 5. Set information for s_{10}.

Step 6: Check for dynamic hazard using all the *P*-sets and *S*-sets generated so far. Notice that the *P*-set, $\{W_{1,8,10}', X_{5,8,10}, X_{2,6,8,10}', Y_{3,6,8,10}\}$, in s_{10} of Table 5 and the *S*-set, $\{W_{1,8}, X_{5,8}', X_{2,6,8}, Y_{3,6,8}'\}$, in s_8 of Table 4 satisfy the corollary for dynamic hazard stated earlier. Thus a dynamic hazard exists.

Step 7: Since $k = 3$ and $\mathbf{S}_3$ has nonzero entries, the program stops.

6 Conclusion

We have shown how to modify a matrix method developed by Heal and Page [1993], for the detection of logic hazards in combinational circuits with EX-OR gates. The matrix method generates sets of variables of all nodes in each gate level of a circuit progressively until it reaches the output of the circuit. In particular, we have defined the rules for the required set generations at input nodes of EX-OR gates. The sets generated are subsequently used to determine the existence of static or dynamic hazards based on McCluskey's work [1986]. Algorithms and examples are given to illustrate set generations and the detection of static and dynamic logic hazards. The results are confirmed by simulations using Xilinx's FPGAs.

References:

[Almaini 1994] Almaini, A.E.A.: "Electronic Logic Systems"; Prentice-Hall, Englewood Cliffs, N.J. (1994).

[Beister 1974] Beister, J.: "A unified approach to combinational hazards"; IEEE Transactions on Computers, C-23, (1974), 566-575.

[Bredeson 1975] Bredeson, J.: "Synthesis of multiple input-change hazard-free combinational switching circuits without feedback"; International Journal of Electronics, 39, (1975), 615-624.

[Eichelburger 1965] Eichelburger, E.B.: "Hazard detection in combinational and sequential switching circuits"; IBM Journal of Research and Development, (1965), 90-99.

[Fleming and Taylor 1987] Fleming, A.R. and Taylor, G.E.: "Matrix methods in the detection and elimination of redundancy in combinational circuits"; in Developments in integrated circuit testing, D.M. Miller (Ed.), Academic Press, (1987).

[Green 1986] Green, D.: "Modern Logic Design"; Addison Wesley, Reading, MA. (1986).

[Heal and Page 1993] Heal, B.W. and Page, R.M.R.: "SIMD matrix methods for detecting hazards in logic circuits"; IEE Proc. - Computers and Digital Techniques, 140, (1993), 201-204.

[McCluskey 1986] McCluskey, E.J.: "Logic Design Principles: with Emphasis on Testable Semicustom circuits"; Prentice-Hall, Englewood Cliffs, N.J. (1986).

[Nelson et al. 1995] Nelson, V.P. Nagle, H.T., Carroll, B.D. and Irwin, J.D.: "Digital Logic Circuit Analysis and Design"; Prentice-Hall, Englewood Cliffs, N.J. (1995).

[Ouyang and Tran 1994] Ouyang, C. and Tran, A.: "Transient analysis and hazard-free design of exclusive-OR switching networks"; IEE Proc. - Computers and Digital Techniques, 141, (1994), 274-280.

[Reddy 1972] Reddy, S.M.: "Easily testable realizations for logic functions"; IEEE Transactions on Computers, C-21, (1972), 1183-1188.

[Sasao and Besslich 1990] Sasao, T. and Besslich, P.: "On the complexity of mod-2 sum PLAs"; IEEE Transactions on Computers, C-39, (1990), 262-266.

[Sasi and Radhakrishnan 1990] Sasi, S. and Radhakrishnan, D.: "Hazards in CMOS circuits"; International Journal of Electronics, 68, (1990), 967-990.

[Tinder 1991] Tinder, R.F.: “Digital Engineering Design: A Modern Approach”; Prentice-Hall, Englewood Cliffs, N.J. (1991).

[Wakerly 1994] Wakerly, J.F.: “Digital Design: Principles and Practices”; Prentice-Hall, Englewood Cliffs, N.J. (1994).