GENERATING FUNCTIONAL DELAY FAULT TESTS FOR NON-SCAN SEQUENTIAL CIRCUITS

Eduardas Bareiša[1], Vacius Jusas[1], Liudas Motiejūnas[2], Rimantas Šeinauskas[1,3]

[1]*Kaunas University of Technology, Software Engineering Department,*
Studentų Str. 50-404, LT-51368 Kaunas, Lithuania
e-mail:eduardas.bareisa@ktu.lt, vacius.jusas@ktu.lt

[2]*Kaunas University of Technology, Multimedia Engineering Department,*
Studentų Str. 50-402, LT-51368 Kaunas, Lithuania
e-mail: liudas.motiejunas@ktu.lt

[3]*Kaunas University of Technology, Information Technology Development Institute,*
Studentų Str. 48A, LT-51368 Kaunas, Lithuania
e-mail: rimantas.seinauskas@ktu.lt

Abstract. The paper presents two functional fault models that are devoted for functional delay test generation for non-scan synchronous sequential circuits. These fault models form one joint functional fault model. The non-scan sequential circuit is represented as the iterative logic array model consisting of k copies of the combinational logic of the circuit. The value k defines the length of clock sequence. The length of clock sequence is determined using the presented functional fault models. The experimental results demonstrate the superiority of the delay test patterns generated at the functional level using the introduced functional fault models against the transition test patterns obtained at the gate level by deterministic test pattern generator. The functional delay test generation method especially is useful for the circuits, when the long test sequences are needed in order to detect transition faults.

Keywords: sequential non-scan circuit, transition fault test, iterative logic array, functional level.

1. Introduction

Testing of sequential circuits can be carried out in either scan mode or non-scan mode. Transition fault testing of sequential circuits has mostly been considered assuming scan that allows a circuit to be tested similar to a combinational one. Two test vectors are applied to detect transition faults, namely v1 and v2. The primary scan-based test techniques are enhanced scan [11], functional justification also called broadside test [20], and scan shifting also called skewed load [15]. All of these techniques use slow and rated clock periods. Slow clock period is used for generation and application of vector v1, as well as for generation of vector v2. The rated clock period is used for application of vector v2 only.

Many sequences can be applied for testing scan based circuits, which cannot be possible during its normal operation. This leads to over-testing of the circuit, which not only increases the test application time, but could also result in loss of yield [19]. Over-testing may become more important when transition faults are targeted compared to over-testing of stuck-at faults [8].

Testing of a delay fault in a non-scan sequential circuit requires more than two vectors. Two methodologies can be applied: variable clock [13] and rated clock [6]. In the variable clock non-scan sequential test methodology, the vector pair should be like the one used in the scan based test methodology. But, the vector v1 should be generated by a set of vectors starting at some initial state. This set is called a justification sequence. If the destination path is a flip-flop, then the state should be propagated to some primary outputs. This part of the test is called a propagation sequence. The slow clock is used for justification and propagation sequences. Thus, only one vector v2 in the entire test sequence uses the rated clock.

The rated-clock non-scan sequential test is the most natural form of the test. All the vectors, either functional or those generated to cover any types of faults, are applied at the rated clock. The variable-clock test is always possible for a fault that is testable by a rated-clock test [13]. However, some variable-clock tests may cover paths that are impossible to activate in the normal rated-clock function.

Under scan-based tests, transition faults are associated with an extra delay that is large enough to cause the delay of any path through the fault site to exceed the clock period [5]. Beyond this assumption, the specific delay size is not important. When non-scan test sequences are applied at-speed, a faulty line must be considered under multiple consecutive fast clock cycles. In this case, it becomes necessary to consider fault sizes measured in numbers of clock periods in order to determine the value of a faulty line. In the transition fault model introduced in [9], each transition fault in the combinational logic of the circuit defines several faults with different extra delays. The transition fault with a given extra delay of l clock periods is referred to as an l-transition fault. An alternative model, which is called an unspecified transition fault, to the one of [9] was introduced in [16]. This model attempts to encompass all the possible sizes of a transition fault in one fault. Under an unspecified transition fault, an unspecified value is introduced at the fault site in the faulty circuit when the fault is activated or when a fault effect is propagated from a previous time unit. Fault detection potentially occurs when an unspecified value reaches a primary output. But the simulation of unspecified values using three-value logic has an inherent loss of the accuracy [16].

Experimental results reported in [9] and [17] indicate that one-transition faults are the hardest to detect. Moreover, tests for one-transition faults can detect most of the l-transition faults for $l > 1$. Therefore, it is possible to conclude that there is no need to construct transition tests for l-transition fault, where $l > 1$. This conclusion is supported by a new model of transition faults, which is introduced in [18]. The model, which is called double-single stuck-at fault, requires the activation of single stuck-at faults with opposite stuck-at values on the same line at consecutive time units. In addition, it requires the detection of both faults (as single faults) at the same or later time units. The application of double-single stuck-at fault model is demonstrated for the transformation of a test sequence for single stuck-at faults into a test sequence for detecting transition faults only. The use of this model for deterministic sequential test generation is not specified.

The transition fault test for non-scan sequential circuits could be constructed at the functional level using the software prototype model, as well [3, 12]. Kang et al. [12] suggested the input/output transition (TRIO) fault model for functional test selection at the register-transfer level (RTL). It is defined with respect to the primary inputs, primary outputs, and state variable of the module. But this model is approximate due to the following reasons: 1) it does not stipulate toggle propagation all the way to the primary outputs; 2) the evaluation of the transition at the output, which depends on multiple input transitions, is too much optimistic. Therefore, the presented experimental results demonstrate quite a large loss of transition fault coverage of the initial test pool. For the circuits' s1196 and s1238, the loss is even 14.99%.

Bareiša et al. [3] introduced three different new fault models: the functional clock at-speed (FCaS), the functional clock static-based (FCS), and the functional clock delay (FCD). According to the proposed models, the functional faults are considered on the primary inputs, primary outputs and the state bits of the model. Bareiša et al. [3] presented the bare ideas of the models only. No implementation details were presented. The experiments were carried out with FCS model only. The important part of the test generation process for non-scan sequential circuits – the determination of the length of the clock sequence was left without attention. Therefore, in this paper, we are going to elaborate FCaS fault model, which now seems to be the most appropriate, and to define in details, the whole test generation process using functional fault models.

The object of the paper is to present the functional delay fault test generation process using the software prototype model. The rest of the paper is organized as follows. We present the functional fault models in Section 2. We introduce the test generation process in Section 3. We report the results of the experiment in Section 4. We finish with conclusions in Section 5.

2. Fault model

A synchronous sequential circuit can be transformed into an iterative logic array [14]. The iterative logic array model of the synchronous sequential circuit consists of duplicated copies of the combinational logic of the circuit, called time frames, as shown in Figure 1. The iterative logic array model for the circuit is expanded for k time frames. The vertical inputs of a combinational cell are primary inputs and the vertical outputs are primary outputs of the sequential circuit; the horizontal inputs are the present state bits and the horizontal outputs are the next state bits.
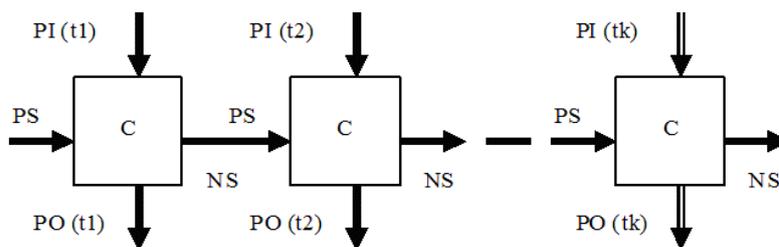
Figure 1. The iterative logic array model

The length of clock sequence k defines the number of the cells (time frames) in the iterative logic array. In such a model, the number of primary inputs is multiplied by k, the number of primary outputs is multiplied by k, the number of previous state bits, which are considered as the primary inputs, is multiplied by k, the number of next state bits, which are considered as the primary outputs, is multiplied by k. We obtain the model of the sequential circuit, which is expanded quite a lot, but all the control of the model is included into the interface.

Let a generic cell of the iterative logic array model have a set of primary inputs $X = \{x_1, ..., x_i, ..., x_n\}$, a set of primary outputs $Y = \{y_1, ..., y_j, ..., y_m\}$, a set of bits of previous state $Q = \{q_1, ..., q_j, ..., q_v\}$, and a set of bits of next state $P = \{p_1, ..., p_j, ..., p_v\}$. The number v is the same for the bits of previous and next states. Therefore, the input stimulus has $n+v$ signal values, and the output stimulus has $m+v$ signal values. We do not relate the inputs and outputs to the time frame, but we associate the signal values to the time frame when we consider the input stimulus and output responses. We denote the complete input stimulus of the cell of the time frame t by $S^t = <s^t_1, ..., s^t_i, ..., s^t_{n+v}>$. The complete output response captured on the outputs of the cell of the time frame t is $R^t = <r^t_1, ..., r^t_i, ..., r^t_{m+v}>$. When we refer to the input stimulus of the whole iterative logic array, we do not use the upper index t.

We define the functional faults for one generic cell, but they will be applied for every cell in the iterative logic array model. Nevertheless, the detectability of the functional faults will be stored for one generic cell only. Such a mode of storing information requires the definition of corresponding functional fault models.

The functional faults are separated into two groups: primary and secondary. The definitions similar to the description presented in [2] are introduced.

Definition 1. The primary functional fault is a pair of stuck-at faults (x_i^f, y_j^h), f=0,1, h=0,1.

Definition 2. The secondary functional fault is a pair of stuck-at faults in one of two different forms:

a) (x_i^f, p_j^h), f=0,1, h=0,1;
b) (q_i^f, y_j^h), f=0,1, h=0,1.

These two functional fault models are not replacing each other, because they cover the different relationships of the generic cell. The secondary functional fault has to be used as the addition to the primary functional fault. We introduced the secondary fault in order to avoid the problem of early saturation that is characteristic for the primary functional fault. The early saturation happens due to the following reasons:

1. Usually the primary input and the primary output are connected by the number of different paths [4].

2. Storing of the functional faults for one generic cell overlays faults of every cell in the iterative logic array.

Now, we are concerned how to use these functional fault models for the detection of transition faults. Remember the description of the detectability of the functional fault [2], which we present here as a definition.

Definition 3. The functional fault (x_i^f, y_j^h) is detected by test stimulus S under the following conditions:

1. The test stimulus S detects the single fault x_i stuck-at f.

2. The fault free value of output y_j under S is $\bar{h}$.

3. In the presence of x_i stuck-at f, the value of output y_j is h.

Such a definition is valid for the detection of stuck-at faults. In order to adopt Definition 3 for detection of delay faults in iterative logic array model we have to take into account the following features:

1. The iterative logic array model consists of k cells; meanwhile the functional faults described in [2] were defined for a single combinational cell.

2. The functional faults are defined for a one generic cell.

3. The fault effect can start at the inputs of the cell t and it can be observed at the outputs of the same cell t or at the outputs of the cells that are located further in the chain of the cells.

4. The stuck-at faults can be injected at the inputs of all the cells and the responses can be observed at the outputs of all the cells.

5. The bits of previous and next state are not real primary inputs and outputs.

6. The delay fault has to be detected. In order to detect the delay fault a transition has to start at the fault site.

Bearing in mind the above listed features, we introduce the following definition that names the necessary conditions for detection of transition faults using the model of primary functional fault.

Definition 4. The primary functional fault (x_i^f, y_j^h) is detected by test stimulus S under the following conditions:

1. The test stimulus S detects the single fault x_i stuck-at f on the input of the cell t.

2. The fault free value under S at the output y_j of the cell t or the cells $t+1$, $t+2$, …, k is $\bar{h}$.

3. In the presence of x_i stuck-at f on the input of the cell t, the value at the output y_j of the cell t or the cells $t+1$, $t+2$, …, k is h.

4. The fault free value under S at the input x_i of the cell t-1 is f.

The last condition of Definition 4 guarantees that the transition starts at the input x_i of the cell t. The first three conditions ensure that the sensitive path exists between the input x_i of the cell t and the output y_j, which can be an output of one of the following cells t, $t+1$, …, k.

The secondary functional fault (x_i^f, p_j^h) does not relate the primary input to the primary output.

Consequently, it alone cannot ensure the propagation of fault effect from the primary input to the primary output. The additional functional fault has to be linked into the chain with secondary functional fault. Now, we can formulate a definition that determines the necessary conditions for detection of transition faults using the secondary functional fault (x_i^f, p_j^h).

Definition 5. The secondary functional fault (x_i^f, p_j^h) is detected by test stimulus S under the following conditions:

1. The functional fault satisfies the conditions of Definition 4 and it is detected at the output p_j of cell t.

2. The functional fault (q_i^f, y_j^h), where q_i denotes the input of the cell $t+1$ directly connected to the output p_j of the cell t, and $p_j^h = q_i^f$, has to be detected according to the conditions of Definition 4, except the fourth condition.

The secondary functional fault (q_i^f, y_j^h) relates the state bit to the primary output. This fault allows modeling the transition fault that starts at the state bit and propagates to the primary output.

Definition 6. The secondary functional fault (q_i^f, y_j^h) is detected by test stimulus S under the following conditions:

1. The test stimulus S detects the single fault q_i stuck-at f on the previous state input of the cell t.

2. The fault free value under S at the output y_j of the cell t or the cells $t+1, t+2, \ldots, k$ is $\bar{h}$.

3. In the presence of q_i stuck-at f on the previous state input of the cell t, the value at the output y_j of the cell t or the cells $t+1, t+2, \ldots, k$ is h.

4. The fault free value under S at the previous state input q_i of the cell $t-1$ is f.

The delay test generation using the secondary functional faults allows sensitizing the paths connecting every bit of state to the primary output.

The detection of the functional delay faults can be represented by the detection matrix $D=\|d_{a,b}\|_{2(n+v),2m}$, where index a is used to denote the inputs of the cell, and index b is used to denote the outputs of the cell. The bits of next state are not represented in the matrix, because the corresponding functional faults are not considered. The entry of the matrix $d_{a,b}:=1$ if the corresponding functional delay fault is detected, $d_{a,b}:=0$ – in the opposite case. Each input/output pair (i, j) is associated with four entries of the matrix $d_{2i-1,2j-1}, d_{2i-1,2j}, d_{2i,2j-1}, d_{2i,2j}$ that correspond to the primary functional delay faults $(x_i^0, y_j^0), (x_i^0, y_j^1), (x_i^1, y_j^0), (x_i^1, y_j^1)$, when $i=1,\ldots, n$, and $j=1 \ldots, ,m.$, and the secondary functional faults are represented by the pairs $(x_i^0, p_j^0), (x_i^0, p_j^1), (x_i^1, p_j^0), (x_i^1, p_j^1)$, when $i=1,\ldots, n$, and $j=m+1 \ldots, ,m+v.$. The entry of the matrix $d_{2i-1,2j-1}$ is set to 1 if the primary functional delay fault (x_i^0, y_j^0) is detected. That corresponds to the situation where the transition $0\rightarrow1$ is on the input i, the transition $0\rightarrow1$ is on the output j, and the blocked transition on the input disables the transition on the output. The entry of the matrix $d_{2i-1,2j}$ is set to 1 if the primary functional delay fault (x_i^0, y_j^1) is detected. That corresponds to the situation where the transition $0\rightarrow1$ is on the input i, the transition $1\rightarrow0$ is on the output j, and the blocked transition on the input disables the transition on the output. The entry of the matrix $d_{2i,2j-1}$ is set to 1 if the primary functional delay fault (x_i^1, y_j^0) is detected. That corresponds to the situation where the transition $1\rightarrow0$ is on the input i, the transition $0\rightarrow1$ is on the output j, and the blocked transition on the input disables the transition on the output. The entry of the matrix $d_{2i,2j}$ is set to 1 if the primary functional delay fault (x_i^1, y_j^1) is detected. That corresponds to the situation where the transition $1\rightarrow0$ is on the input i, the transition $1\rightarrow0$ is on the output j, and the blocked transition on the input disables the transition on the output. In the same way, the detection of the secondary functional faults is labeled when they are detected according to Definition 5.

3. Test generation process

Delay test generation is accomplished at the functional level. The model of the circuit has to be described in a high level description code, which is termed a software prototype. Therefore, it can be presented in the form of a high level programming language, behavioural VHDL or Verilog description code. But the reality is such that the models of the circuits usually are available in the RTL description code, for example ITC'99 benchmark suite [10]. Such models have to be lifted up into the algorithmic level of the description. In order to achieve this goal there are several ways: 1) to write the model in C programming language; 2) to translate from VHDL or Verilog RTL code to the code in C programming language; 3) to translate from VHDL or Verilog structural code to the code in C programming language. There are possible other alternatives, but we did not consider them. We have tried to write the models in C programming language for all the benchmarks from ITC'99 benchmark suite. But we did not achieve our goal, because for all the models practically it is not possible to ensure the adequacy. The second way, the most attractive and reliable one, was eliminated as not possible for two reasons: 1) it is difficult to think of the rules that would allow to convert several parallel processes into sequence of the operators in C programming language; 2) such a way contradicts to the whole design process, which flows from algorithmic level to more detailed RTL. The third way looks little bit strange, but the synthesized structural descriptions are available for all the benchmarks. We have written a translator from Verilog structural description code into code in the C programming language. The rules of the translation are very simple, because every Verilog primitive (and, or, not, nand, nor) can be substituted by appropriate operator of C programming language. Such a model has a single deficiency only – it is very

large for large circuits. Therefore, the productivity of the test generation program suffers quite a lot.

Usually the reset and clock signals are present in the RTL description code. The values of the reset and clock signals change according to the regular law. Therefore, these inputs have to be excluded from the consideration. The values according to regularities of these inputs have to be supplied later when the final test is obtained for the sequential circuit.

In order to use the introduced fault models, the state bits have to be extracted from the model of the circuit. In a high level description code, the state bits are represented by the variables. The declared type of the variable determines the number of bits required for the variable. But not all the variables represent the state; some of them are used for temporary storage of the values only. The careful analysis of the code is needed in order to determine which variables are temporary. Synthesis of the code could aid to resolve this problem. Consider an example. Let us use the RTL code of circuit b01 from ITC'99 benchmark suite presented in VHDL hardware description language. We find a single declared variable in the code:

```
variable stato: integer range 7 downto 0;
```

The knowledge of VHDL language allows determining that three state bits are required for the variable `stato`. Let us examine the synthesized description of circuit b01. We find five flip-flops. In order to find out the problem of difference between the number of state bits and the number of flip-flops we examine the synthesized code of b01. We discover that two additional flip-flops are connected to two primary outputs of the circuit. Such flip-flops form a buffer zone. The buffer zone can be formed on the primary inputs as well. But the flip-flops of the buffer zone can be neglected, because they are used for the temporary storage of the values only. Therefore, the initial determination that three state bits are required for the circuit b01 was correct. We could say in advance that all the circuits from benchmark suite ITC'99 have the buffer zone of flip-flops at the primary outputs, except the circuit b05.

The parameters of circuits from the benchmark suite ITC'99 are presented in Table 1. We have to pay attention to the fact that the number of inputs of fault model does not count the reset and clock signals that are present in all the circuits. Analysis of the VHDL code of the benchmarks presented in Table 1 revealed that the code of circuits' b04, b05, b08, b12, b14 has temporary variables. In the code of circuits' b07 and b10, we learnt that some bits of declared variables are never used. Therefore, the number of state bits according to our calculation is more than the number of flip-flops minus the number of primary outputs that are represented by the flip-flops of the buffer zone.

Now we present a delay test generation process. The delay test generation process consists of two stages: determination of length of clock sequence, and test pattern generation. The first stage is a very

important stage, because the non-scan sequential circuit is represented by the iterative logic array containing k combinational copies of the sequential circuit. In other words, k denotes the length of clock sequence. Every sequential circuit especially at the algorithmic level can be represented as a finite state machine. The finite state machine is always synchronized by clock sequence of some defined length. If the length of clock sequence is too short, some states will not be visited, and the corresponding delay faults will not be detected. If the length of clock sequence is too long, some states will be visited repeatedly but that will not sensitize the new paths, and the new faults will not be detected. Too long clock sequence increases the number of test patterns in the test sequence quite substantially but without a necessity. Therefore, the number of copies k directly influences the success of test pattern generation.

Table 1. Parameters of circuits

	Circuit				Fault model	
Name	In-puts	Out-puts	State bits	Flip-flops	Inputs	Out-puts
b01	4	2	3	5	5	5
b02	3	1	3	4	4	4
b03	6	4	26	30	30	30
b04	13	8	58	66	69	66
b05	3	36	34	34	35	70
b06	4	6	3	9	5	9
b07	3	8	43	49	44	51
b08	11	4	17	21	13	21
b09	3	1	27	28	28	28
b10	13	6	14	17	25	20
b11	9	6	25	31	32	31
b12	7	6	115	121	120	121
b13	12	10	43	53	53	53
b14	34	54	191	245	223	245

In order to determine the length of clock sequence we use the fault models presented in Section 2. The secondary functional fault (x_i^f, p_j^h) model shows the ability to control and to observe the state bit. Of course, we understand that all the state bits have to be controlled by the values on the primary inputs. Therefore, the secondary functional fault (x_i^f, p_j^h) model serves as the first criterion in choosing the correct length of clock sequence. We count the number of uncontrollable state bits according to the secondary functional fault (x_i^f, p_j^h) model. The goal is that this number would become equal to zero. The increase of the length of clock sequence allows us to converge to this goal. This goal is not always reachable. Sometimes, we increase the length of clock sequence quite substantially to several thousands, but some state bits still remain uncontrollable. When we reach the goal or we see that it is not possible to control all the state bits

with acceptable length of clock sequence, the primary functional fault model is used as the second criterion in choosing the correct length of clock sequence. Then we count the number of detected functional faults according to both criteria. We never know what the number of detectable faults is. Therefore, the goal is to reach the number of detected functional faults as larger as possible. We stop the increase of the length of clock sequence, when the number of detected functional faults does not augment or the growth is very small.

To start the delay test generation, the circuit is assumed to be initialized to state 0 before the application of the input sequence. If the circuit has a *reset* input, then it can be set to state 0. If the circuit has no the reset input, the synchronizing sequence could be used, which transfers the circuit to known initial state that could be different from the state 0. For example, all the benchmarks from ITC'99 suite have the reset input; meanwhile the benchmarks from ISCAS'89 suite have no reset input [7]. The problem of the generation from the initial state, which is not the state 0, is out of scope of this paper.

When the length of clock sequence, which defines the number of the cells in the iterative logic array model of the circuit, is determined, the next step is a delay test pattern generation. The generation is implemented according to the iterative logic array model of the circuit using the models of the primary functional fault and the secondary functional fault (q_i^f, y_j^h). The secondary functional fault (x_i^f, p_j^h) is not used for the delay test pattern generation. The influence of the values on the primary inputs to the values on the primary outputs and the influence of the values on previous state bits to the values on the primary outputs are considered only.

The circuit is considered to be set to the state 0. Random values are generated on the primary inputs. Simulation is carried out of the first cell of the iterative logic array. Simulation defines the values on the primary outputs and the values of the next state, which become the previous state values for the next cell of the iterative logic array. The primary functional fault and the secondary functional fault (q_i^f, y_j^h) are simulated on every cell of the iterative logic array. Definition 4 and Definition 6 state the conditions of the detectability of the functional faults. The detected functional faults are labeled in the detection matrix D. The test generation process should stop when all the simulated functional faults are detected, but the number of testable functional faults is unknown. The solution to the problem of stopping the functional delay test generation using the detection matrix is presented in [1].

4. Experimental results

The experiments were carried out on the circuits of the benchmark suite ITC'99. We report the detailed process of the determination of the length of clock

sequence for the circuit b01 in Table 2. The length of clock sequence is directly related to the functioning of the circuit. In order to easier understand the whole process of the determination of the length of clock sequence we present the state transition graph of circuit b01 in Figure 2. The state transition graph has 8 states. The names for the states are given according to VHDL model of circuit b01. The circuit has 2 primary inputs (the reset and clock inputs are not counted). The values are shown only on those edges, where equal values are required on both inputs. These values indicate that the path traversing the states of the circuit will be chosen more likely through the edges that do not have the values, because the generation of the values is random. For example, the edge, which connects the vertices a and f, is labeled by the controlling value 11. The transition by the other edge, which connects the vertices a and b, is controlled by the following three combinations of values: 00, 01, 10. Therefore, this transition will happen 3 times more likely. Such a consideration allows us to determine the proper length of clock sequence.

Table 2. Determination of length of clock sequence for b01

Length of clock se-quence	Number of uncontrol-lable state bits	Number of detec-ted functio-nal faults	Number of test sub-sequenc es	Fault cove-rage at gate level (%)
4	2	15	–	–
8	0	44	–	–
16	0	49	–	–
32	0	49	–	–
12	0	42	–	–
10	0	44	–	–
9	0	49	10	97.37
			12	96.99
15	0	49	8	93.61
			6	96.24
14	0	45	–	–

As one can see in Table 2, we start the generation with the length of clock sequence, which equals to 4. Knowing the function of the circuit, it was possible to predict that such a length of clock sequence will leave some state bits uncontrollable. This value was chosen in order to show that the model allows counting up the uncontrollable state bits. Then, we double the value of the length of clock sequence. The law of doubling is used always in the search of the proper length of clock sequence. In this search, we find two least values that would fit for the proper length of clock sequence. Value 9 is found only, because we know the functioning of the circuit. The algorithm indicates the value 15; the analysis of the state transition graph presented in Figure2 suggests the value 9, therefore, the decision was made to generate test sequences for

both lengths of clock sequences. In order to reduce the factor of randomness the generation was carried out two times. The last two columns show the results of these generations. We did not use the term test subsequence, which is shown in the fourth column in the text before. The test subsequence is a sequence of input patterns, which corresponds to one clock se-

quence. Every test subsequence starts with reset test pattern.

As the results indicate, the analysis of the state transition graph was performed appropriately – the length of clock sequence should be 9.
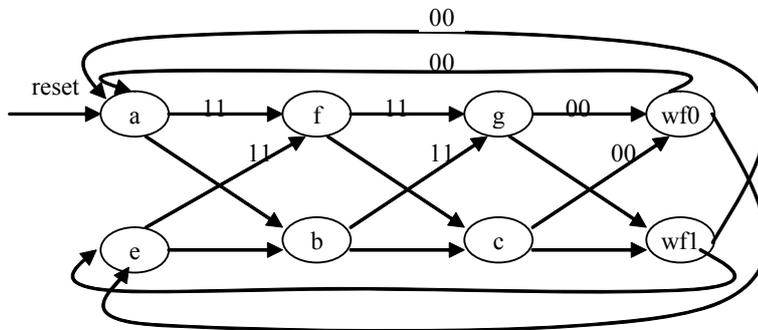
Figure 2. State transition graph of circuit b01

The similar process of the search for the proper length of clock sequence was carried out for all the circuits presented in [1], but we do not provide the details. The stress is made now on the functional delay test pattern generation. The results are reported in the first three columns following the column of circuits' names. We would like to pay attention that the fault coverage of the functional delay test patterns was measured at the gate level.

Table 3. Functional delay test patterns

Circuit	Functional delay test patterns			Fault coverage (%)	
	Length of clock sequence	Number of test sub-sequences	Fault coverage (%)	Tetra-MAX	[18] method
b01	9	10	97.37	97.37	-
b02	9	6	86.36	86.36	-
b03	9	85	56.09	52.15	55.73
b04	17	166	83.23	82.64	79.03
b09	64	14	60.14	62.70	65.93
b10	30	151	79.23	77.48	76.55
b11	2160	11	78.82	50.14	79.13
b12	480	27	34.33	6.6	-
b13	2160	92	63.43	21.29	-
b14	240	444	76.76	32.37	-

The transition test patterns were generated at the gate level by TetraMAX program. The results are presented in the penultimate column. The last column reports the results obtained in [18]. These two columns are provided for comparison purposes.

We see that the fault coverage of functional delay test patterns is larger or equal in comparison with the fault coverage of transition test patterns generated by TetraMAX for all the circuits, except the circuit b09.

Especially good results are obtained for the circuits' b11, b12, b13 and b14, where the long test sequences are needed in order to detect transition faults. We could confess that the thorough work was needed in order to select the proper length of clock sequence, because the process is not automatic yet. We see that the length of clock sequence is very large for circuits' b11, b13 and the lengths are occasionally equal.

The obtained results of functional delay test generation can be compared with the results provided in [18], where the double-single fault model was used for transition test generation. Our method of functional delay test generation allows obtaining better results for the following circuits: b03, b04, and b10. Our method looses for the circuit b09 and it obtains almost the same fault coverage for the circuit b11.

Our method of functional delay test generation has else one advantage over the transition test generated by TetraMAX – shorter time of test generation. For example, TetraMAX generates the transition test patterns an hour and 56 minutes for the circuit b09, an hour 45 minutes – for the circuit b10. This amount of the time is already quite significant, but it increases to half a day for the larger circuits like b11, b12, b13, and b14. The test generation according to our method obtains the results within the seconds for the smaller circuits (b03, b04, b09, b10) and it takes some minutes for the larger circuits (b11, b12, b13, b14). The time of test generation in [18] was not provided.

5. Conclusions

We presented two functional fault models that are devoted for functional delay test generation for non-scan synchronous sequential circuits, namely the primary functional fault model and the secondary functional fault model. The first model deals with the stuck-at faults on the primary inputs and primary

outputs. The second model, which is used as the addition to the first model, deals with the stuck-at faults on the primary inputs, state bits and primary outputs. The circuit is represented as the iterative logic array model, consisting of k copies of the combinational logic of the circuit. The value k defines the length of clock sequence. The method that allows determining the length of clock sequence was presented.

The obtained results show that the introduced delay test generation method, using the presented functional fault models, outperforms by the fault coverage the transition test patterns obtained at the gate level by deterministic test pattern generator. The introduced delay test generation method obtains especially good quality results for the circuits, when the long test sequences are needed.

References

[1] **E. Bareisa, V. Jusas, K. Motiejunas, R. Seinauskas**. Test Generation at the Algorithm-Level for Gate-Level Fault Coverage. *Microelectronics Reliability*, 2008, *Vol*.48, *Issue* 7, 1093-1101.

[2] **E. Bareisa, V. Jusas, K. Motiejunas, R. Seinauskas**. Functional Delay Test Generation Based on Software Prototype. *Microelectronics Reliability*, 2009, *Vol*.49, *Issue* 12, 1578-1585.

[3] **E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas**. Functional Delay Clock Fault Models. *Information Technology and Control*, Kaunas, Technologija, 2008, *Vol*.37, *No*.1, 12 - 18.

[4] **E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas**. On the Enrichment of Functional Delay Fault Tests. *Information Technology and Control, Kaunas, Technologija*, 2009, *Vol*.38, *No*.3, 208 – 216.

[5] **Z. Barzilai, B. Rosen.** Comparison of AC Self-Testing Procedures. *Proceedings of the IEEE International Test Conference*, 1983, 89–94.

[6] **S. Bose, V.D. Agrawal**. Sequential Logic Path Delay Test Generation by Symbolic Analysis. *Proceedings of the 4th Asian Test Symposium, Nov*. 1995, 353-359.

[7] **F. Brglez, D. Bryan, K. Kozminski**. Combinatorial Profiles of Sequential Benchmark Circuits. *Proceedings of IEEE International. Symposium on Circuits and Systems*, 1989, 1929-1934.

[8] **G. Chen, S.M. Reddy, I. Pomeranz.** Procedures for Identifying Untestable and Redundant Transition Faults in Synchronous Sequential Circuits. *Proceedings of the 21st International Conference on Computer Design* (*ICCD'*03), 2003, 36-41.

[9] **K.-T. Cheng.** Transition Fault Testing for Sequential Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol*.12, *No*.12, *Dec*. 1993, 1971–1983.

[10] **F. Corno, M. Sonza Reorda, G. Squillero**. RT-Level ITC'99 Benchmarks and First ATPG Results, *IEEE Design and Test of Computers, Vol*.17, *No*.3, July-Sept. 2000 , 44-53.

[11] **S. Dasgupta, R.G. Walther, T.W. Williams**, **E.B. Eichelberger**. An Enhancement to LSSD and Some Applications of LSSD in Reliability, Availability and Serviceability. *Proceedings of the International Symposium on Fault Tolerant Computing*, 1981, 880-885.

[12] **J. Kang, S.C. Seth, V. Gangaram**. Efficient RTL Coverage Metric for Functional Test Selection. *Proceedings of the 25th IEEE VLSI Test Symposium* (VTS'07), 2007, 318-324.

[13] **S. Majumder, V.D. Agrawal, M.L. Bushnell**. Path Delay Testing: Variable-clock versus Rated-clock. *Proceedings of the 11th International Conference on VLSI Design, Jan*. 1998, 470–475.

[14] **P. Muth.** A nine-valued circuit model for test generation. *IEEE Transactions on Computers, Vol*.25, *No*.6, *June* 1976, 630-636.

[15] **S. Patil, J. Savir**. Skewed-Load Transition Test: Part II, Coverage, *Proceedings of the IEEE International Test Conference*, 1992, 714-722.

[16] **I. Pomeranz, S.M. Reddy**. A Delay Fault Model for at-Speed Fault Simulation and Test Generation. *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Nov*. 2006, 89–95.

[17] **I. Pomeranz, S.M. Reddy**. Unspecified Transition Faults: A Transition Fault Model for At-Speed Fault Simulation and Test Generation. *IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems, Vol*.27, *No*.1, *January* 2008, 137-146.

[18] **I. Pomeranz, S.M. Reddy**. Double–Single Stuck-at Faults: A Delay Fault Model for Synchronous Sequential Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol*.28, *No*.3, *March* 2009, 426-432.

[19] **J. Rearick.** Too much Delay Fault Coverage is a Bad Thing. *Proceedings of the IEEE International Test Conference*, 2001, 624-633.

[20] **J. Savir, S. Patil**. Broad-Side Delay Test. *IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems, Vol*.13, *No*.8, *August* 1994, 1057-1064