TEST GENERATION FOR DELAY FAULTS USING STUCK-AT-FAULT TEST SET

Chi-Chang Liaw, Stephen Y. H. Su and Yashwant K. Malaiya

Research Group on Design Automation and Fault-tolerant Computing
School of Advanced Technology
State University of New York - Binghamton

ABSTRACT

Delay faults are referred to the out-of-specification path delays which result in unstabilized or incorrect circuit behaviors. The abnormal delay in components can cause intermittent (transient) faults which are responsible for the most faults in the digital equipment in the field. When one uses design-for-testability discipline, such as the level-sensitive scan design (LSSD), testing for the timing failures is required mainly for delay faults which exceed the maximum allowable propagation time along signal paths in the combinational network part of a digital network. The idea of path sensitization is used for finding tests for delay faults. In this paper, we analyze the relationships between stuck-at-fault tests and delay tests (tests for delay faults). It is shown that with only a minor modification, many stuck-at-fault tests can be adopted as delay tests and such delay tests may simultaneously detect stuck-at-faults and delay faults along more than one path. Consequently, we propose to start delay-test generation with a given stuck-at-fault test set which is usually available. A selection process is applied to select tests according to certain criteria. Tests for the uncovered delay faults are then generated by using currently available methods. An experimental selector has been implemented in PL/I on an Itel AS-6 computer. The test patterns generated for an off-the-shelf 4-bit ALU chip shows a good coverage is obtained by using only a small number of tests.

INTRODUCTION

Testing to verify the timing performance of manufactured logic circuits is considered to be more difficult than testing for logic verification. In conventional designs, timing failures may appear in one or a combination of several forms, such as hazards, races or exceedingly long or short propagation delays. Solutions for testing all such timing failures have not been satisfactory, and the problem becomes even more serious when more elements are fabricated into one chip. To lessen the LSI (large scale integration) testing problems, some design disciplines have been adopted. For example, the level-sensitive scan design discipline (LSSD)[1] produces a logical structure of combinational circuits separated by banks of latches. In the testing mode, the latches can be structured to form shift registers. Test patterns and test results can be scanned in and out through the shift register structures and accessed at the external pins. It has been observed that, under this discipline, testing for timing failures is required only for path delays which exceed the maximum allowable propagation time in the combinational circuits. (The maximum time is usually determined by the operating clock rate.) In this paper, we shall consider this type of timing failure, and call them delay faults. A path may exhibit excessive delay in responding to a rising input or to a falling input, or both. Each path therefore has to be examined separately for rising and falling input.

A test for a delay fault involves (1) sensitizing the input-output path, (2) making a transition at the input, and (3) latching or strobing the output after a period of delay. Two patterns are required in each test. Two different approaches for testing can be found in the literature: The first approach[2] assumes the period between transition and latching (or strobing) is equal to the maximum allowable propagation time (in addition to the set-up and propagation delays of the latches[3]). The outcome of a test is then binary: succeed or fail. Since the number of paths in a practical circuit is prohibitively large, completely testing all delay faults becomes economically infeasible. The criterion used instead is to let each element input or output be tested in two paths as disjoint as possible, by selecting the longest and shortest delay paths that can be sensitized. A test generator selects paths according to this criterion and derives test patterns by path sensitizing techniques. The second approach[4] assumes the period between input transition and output latching is adjustable. Therefore, after applying and adjusting a test a number of times, the propagation delay can be measured. It has been shown that the propagation delays of all paths, which can be sensitized in single-path (or one-dimensional) propagations, can be computed from the delays of some properly chosen subset of paths. An almost-complete delay testing can be achieved by measuring these path delays and a post-test calculation.

This work is supported by the National Science Foundation, Division of Mathematical and Computer Science under Grant Number MCS 7824323.

0-8186-6617-X/94 $04.00 © 1980 IEEE

Although this approach yields a very high quality of testing, the required time for measuring and the post-test calculation could be prohibitive.

In the next section, we shall introduce a new approach. Like the first approach, it also assumes a fixed period between transition and latching. It starts with a given stuck-at-fault test set, selects test patterns according to some criteria, and then modifies the patterns into delay tests. (The delay faults not covered by such delay tests can be identified, and tested by other available methods.) A delay test obtained in this way usually includes more than one input transition and tests more than one path at the same time. The foundation for this approach will be presented by two theorems. Then we shall describe an experimental test selector and some experimental data obtained in test generation for an off-the-shelf 4-bit ALU chip, 74S381. The concluding remarks are given in the last section.

TEST GENERATION FOR DELAY FAULTS

Our approach is to use the stuck-at-fault test set for testing delay faults. The reason is the following: Test generation for stuck-at faults in combinational circuits is well known. The test set has to be generated and applied for testing stuck-at faults anyway, so it is available without any additional cost. In addition test generation of both stuck-at faults and delay faults is based principally on path sensitizing techniques. This suggests that some stuck-at-fault tests (one pattern for each test) may be augmented into delay tests (two patterns each) each of which can detect one or more delay faults in addition to the stuck-at faults. For example, in Fig. 1, if input x s-a-1 is sensitizable through one path under a test pattern T (with x = 0), then a two-pattern test T*T, where patterns T* and T are identical except x = 1 in T* and x = 0 in T, can detect the delay fault associated with this path as well as the stuck-at faults originally detected by T. In general, more than one path may be sensitized simultaneously, thus a single delay test may detect several delay faults at the same time. This provides an opportunity to obtain more efficient tests than simply sensitizing one path for each test.

A selection process selects patterns from the stuck-at-fault test set such that the resulting delay tests will test as many delay faults as possible. In that process, information is obtained about which delay faults are covered. The tests for delay faults not covered are then generated by other currently available methods such as the scheme used in the first approach.

To make our later discussions more specific, we present a simplified model for the combinational circuit-under-test and its testing environment as shown in Fig. 2. All inputs and outputs of the combinational circuit are latched. Some latches are imbedded in the chip, while others are externally provided by the tester at the input/output pins of the chip. The internal latches are organized into shift register structures (not shown in the figure). The parallel load clock at input or output register, C1 and C2, may represent a group of clock signals, each of which controls one or more latches in the register. When these clocks are triggered simultaneously during testing, they behave like a single clock. Some assumptions are made here:

(1) The system operates synchronously.

(2) Any pattern can be scanned in the input register R1 and the content of the output register R2 can be scanned out via only the shift register structures.

(3) The parallel load clocks, C1 and C2, are non-overlapping and can be separately controlled. Alternatively, a single clock or overlapping multiple clocks can be used, provided it is ensured that there is no race-around condition.

Note that the above assumptions are valid under the LSSD design discipline. A stuck-at-fault test for CUT (circuit-under-test) can be loaded in R1 via a scan-in operation. Clock C2 is triggered after a sufficient delay. The test result latched in R2 is then observed via a scan-out operation. A delay test for CUT requires transitions to take place at one or more inputs. This can be done by properly arranging logic levels at the inputs of R1 and triggering C1 accordingly. For example, suppose T* and T are the two patterns of a delay test. A scan-in operation is applied first, which loads T* in R1. In the meantime, the previous-stage latches produces T at the inputs of R1. When C1 is triggered, transitions at R1 take place and test pattern T is then loaded into R1. C2 is triggered after the maximum allowable delay which strobes CUT's outputs at R2. A final scan-out operation observes the test result. In practice, if register R1 is composed of segments which can be individually clocked, then only those segments containing changing bits need to be clocked. In our previous single-path example, only those latches containing input x strobed by the same clock need the second test pattern. We shall discuss the subject of simultaneously applying multiple transitions at the end of this section.

We now examine how a delay test can be derived from a stuck-at-fault test. It is assumed that the circuit under consideration is loop-free and composed of AND, OR, NAND, NOR, exclusive-OR gates and inverters. First of all, the set of all nodes sensitizable by the test pattern is obtained by simulation. In order to derive a formula for the propagation delay from a sensitizable input to an output that transitions can reach, we abstract a subnetwork composed of all paths which propagate simple transitions from the input to the output (Fig. 3). The abstracted subnetwork preserves the propagation delays among the input, fan-out and reconvergent points, and the output. Its function is equivalent to a delay element with or without an inverter ($y = x$ or $y = \bar{x}$). Every node in the

abstracted subnetwork has its value equal to x or $\bar{x}$ so that a path containing it can propagate transitions. It is clear that no exclusive-OR gates would appear in the subnetwork, since each input to exclusive-OR must be equal to either x or $\bar{x}$ yielding a constant output 0 or 1. The paths either all have even number of inversions or all have odd number of inversions. A transformation can then be applied on the abstracted subnetwork, using DeMorgan's theorem if required so that the resulting subnetwork has an inversion (if any) only at the output (Fig. 4b). This preserves the structure and delays. Every node in the inversion-free block has the value x. The input-output propagation delay is given by the following recursive rules, D(node p→node q) symbolizes the delay in a subpath between nodes p and q:

(1) If node i feeds node j through a connection, then D(input x→node j) = D(input x→node i) + D (node i→node j).

(2) If node j is the output of an AND gate fed by nodes i_k's, and input x transits from 0 to 1, then the maximum function is introduced as follows:

D(input x→node j) = $\max_k$ [D(input x→node i_k) + D (node i_k → node j)]. If x transits from 1 to 0, then change 'max' into 'min' in the above equation.

(3) If node j is the output of an OR gate fed by nodes i_k's, then interchange 'max' and 'min' in (2).

Example 1 The total delay between input x and output y of the circuit shown in Fig. 3 is computed based on Fig. 4(b) as follows: If input x transits from 0 to 1 (rising input), then

$$
\begin{aligned}
& D(x\to y) \\
&= \max [D(x\to g) + D (g\to y),\ D(x\to h) + D (h\to y)] \\
&= \max [D(x\to e) + D (e\to g) + D (g\to y),\ D(x\to f) + D(f\to h) + D (h\to y)] \\
&= \max \{\min [D(x\to a) + D (a\to e),\ D (x\to b) + D (b\to e)] + D (e\to g) + D (g\to y), \\
&\qquad \max [D(x\to c) + D (c\to f),\ D (x\to d) + D (d\to f)] + D(f\to h) + D (h\to y)\} \\
&= \max \{\min [D(\text{path xaegy}),\ D (\text{path xbegy})], \max [D(\text{path xcfhy}),\ D (\text{path xdfhy})]\} \\
&= \max \{\min [D(\text{path xaegy}),\ D (\text{path xbegy})], D (\text{path xcfhy}),\ D(\text{path xdfhy})\} \qquad (1)
\end{aligned}
$$

If input x transits from 1 to 0 (falling input), then we have

$$
\begin{aligned}
& D(x\to y) \\
&= \min \{\max [D(\text{path xaegy}),\ D(\text{path xbegy})], D (\text{path xcfhy}),\ D (\text{path xdfhy})\} \qquad (2)
\end{aligned}
$$

We can examine which path delays in Example 1 are tested by the transitions. For the case of rising input (see equation 1), if the delay on either or both of the paths xcfhy and xdfhy exceeds the maximum allowable delay, the total delay also exceeds the maximum allowable delay, regardless of the delays on the other two paths. So the path delays on xcfhy and xdfhy are tested. On the other hand, if the path delay on xaegy exceeds the maximum allowable delay but the other three path delays don't, the minimum function produces a value less than the maximum allowable delay yielding a normal total delay. So the path delay on xaegy is not guaranteed testable. The same arguments also apply to the path delay on xbegy in the rising input case and all four path delays in the falling input case (see equation 2). For the conservative purpose, only those faults guaranteed testable can be said to be "detected". So, in Example 1, only two delay faults of rising input associated with paths xcfhy and xdfhy are detected.

The previous discussion can obviously be generalized as the following lemma:

Lemma 1: Given a test pattern, the delay on an input-output path, where a simple transition signal can propagate, is guaranteed testable if and only if the path delay appears as an argument in the maximum function producing the total delay. In order to achieve this, in the abstracted subnetwork after transformation, each gate on the path must introudce a maximum function in computing the total delay.

For example, the delay on path xcfhy or xdfhy appears in equation 1 is testable, and each gate on the path in the subnetwork is AND which introduces a maximum function in the rising input case.

The previous example points out the fact that when a stuck-at-fault test sensitizes multiple paths, there is a possibility that some paths may be masked by other paths. This occurs whenever two (or more) subpaths converge at a gate, and the output of the gate changes immediately with the arrival of the transition through the faster subpath, thus masking any excessive delay that the other subpath might have. Two theorems are presented below which aid delay-test generation.

Theorem 1: Let T be a stuck-at-fault test pattern which sensitizes input x with logic value v (0 or 1). Let delay test T*T be formed, where T* is identical to T, except for input x, which would have value $\bar{v}$. Then all one-dimensional simple transition paths sensitized by T, can be tested for delay faults by using T*T.

Proof: The theorem will be proved by showing that under T*T, none of the delay faults associated with one-dimensional simple transition paths sensitized by T, will be masked.

Every node on a one-dimensional sensitizing path is sensitizable. Consider now the abstracted subnetwork of the combinational circuit with input pattern T. As abstraction and transformation preserve circuit structure, sensitizability is also preserved.

Let us now examine the two alternate possibilities in parallel. Let the value of v be 1 (0). Then all the nodes in the inversion-free block of the abstracted subnetwork after transformation will have logic value 1(0). As all the gates in the one-dimensional sensitizing paths are sensitized with inputs equal to 1 (0), they all must be AND

(OR) gates. Let x change from 0 to 1 (1 to 0). Applying Lemma 1, if the delay associated with any one-dimensional sensitizing path is more than the maximum allowed, then the total delay for transition under T*T will also exceed the maximum. Q.E.D.

Considerable testing time can be saved if, when using a stuck-at-fault test pattern for delay fault testing, two or more inputs can be allowed to change at the same time instead of one at a time. However, when two or more sensitizable inputs are changed, masking may occur if transitions fail to propagate, or turn into glitches (narrow pulses). For example, in Fig. 5(a), both ac and bc are part of one-dimensional sensitizing paths. However, the transitions at the inputs (as shown) will only produce a constant level or a glitch. In Fig. 5(b), even though the two paths indicated are sensitizable, when simultaneous transitions are applied, the output of the AND gate either remains 0 or has a glitch. The delay fault in the upper path is thus masked. Theorem 2 indicates how this problem can be avoided.

Theorem 2: Let a stuck-at-fault test pattern T sensitize a set of inputs (with logic values given by the subpattern p, which is part of T) through several one-dimensional paths. Let a delay test T*T be formed, with T* identical to T except subpattern p is replaced by $\bar{p}$ (with opposite logic values). Then, if the following restrictions are satisfied, T*T will test the delay faults in all paths which are one-dimensionally sensitized by T:

Restrictions: For every gate on a one-dimensional sensitizing path,

(i) any exclusive-OR input, not on the path, does not change its logic value;

(ii) any gate input, not on the path, does not change in the direction opposite to the input on the path.

Proof: The two restrictions above are obvious. We only need to prove that no masking occurs when inputs of any gate (except exclusive-OR) on the paths change in the same direction.

As all gate inputs on the paths are sensitizable under T (which is the second test pattern in delay test T*T), when T is applied, all AND and NAND inputs must be 1's and all OR and NOR inputs must be 0's. Therefore, the changing signals will pass through each gate only when its input signal with the longest delay arrives. Thus no masking can occur. Q.E.D.

Up to now, we have established a good basis for generating delay tests from a stuck-at-fault test. We first find all sensitizable nodes. Based on this information, all one-dimensional sensitizing paths are identified. We then resolve the potential masking problems as indicated before. If masking may happen between a pair of inputs, then one must be held unchanged while the other performs a transition. For example, suppose simultaneous transitions at inputs a and b may result in masking, but simultaneous transitions at inputs a and f, or b and g, will not. Let subscripts '2' indicate the stuck-at-fault test pattern and '1' indicate its complement. A delay test resulting in masking

$$T_1^* = a_1 b_1 c_2 d_2 e_2 f_1 g_1$$

$$T_1 = a_2 b_2 c_2 d_2 e_2 f_2 g_2$$

can be separated into two tests with no masking effects:

$$T_2^* = a_1 b_2 c_2 d_2 e_2 f_1 g_2 \qquad T_3^* = a_2 b_1 c_2 d_2 e_2 f_2 g_1$$

$$T_2 = a_2 b_2 c_2 d_2 e_2 f_2 g_2 \qquad T_3 = a_2 b_2 c_2 d_2 e_2 f_2 g_2$$

The one-dimensional sensitizing paths originating at a, b, f and g are collectively tested by delay tests $T_2^* T_2$ and $T_3^* T_3$.

In general, given a test pattern T, in order to test all the one-dimensional sensitizing paths originating from a set of sensitizable inputs, I, a set of delay tests $\{T^*_{p_i} T \mid \cup_i p_i = I\}$ can be applied (where $T^*_{p_i}$ has the same pattern as T except each input in p_i is different), provided that, for each i, $T^*_{p_i} T$ does not result in masking.

In changing from T* to T, if many variables change their values, in practical circuits, it may not be possible to provide T. Under this case, we can replace T*T by a set of tests, $\{T^*_{p_1} T, T^*_{p_2} T, \ldots, T^*_{p_k} T\}$ such that each $T^*_{p_i} T$ causes some input transitions as long as the entire test set causes all input transitions (without masking one another).

CRITERIA FOR SELECTING DELAY TESTS

A program has been written in PL/I to select a subset of stuck-at-fault test set for testing delay faults. Given a test pattern for stuck-at faults, the program finds all one-dimensional path sensitizable by the test. It also checks and eliminates all possible masking problems due to multiple input transitions. Given any stuck-at-fault test set, the program selects those tests detecting both stuck-at and delay faults and forms a repertoire R.

We implemented three different criteria for selections:

(i) Covering maximum number of delay faults: The goal is to select a subset of R containing near-minimum number of tests for detecting all the delay faults coverable by R. To obtain a near-minimum solution, the tests in R are sorted in the descending order in terms of the number of delay faults covered by each test. The selection process then selects tests from the top of the

list until all delay faults coverable by R are found.

(ii) Covering maximum number of gate-output pairs: The above criterion although gives the best result on what one can obtain from a given stuck-at-fault test set, the resulting delay test set tends to be large. If delay failures are essentially found in gates, then it is not necessary to test all paths. In most current technologies, the likely causes for delay faults are associated with gate outputs rather than gate inputs or connections among gates, it is justifiable to consider only the delay failures at gate outputs. Our second criterion is to select a subset of R containing near-minimum number of tests for testing all the gates and their outputs (for both 0 and 1) coverable by R. The selection process again selects tests from the top of the sorted list until all gate-output pairs coverable by R are found.

(iii) Covering maximum number of gate-output pairs in two crossing paths: If a gate is tested only in one path, its delay failure may possibly be masked by other gates on the path which have less-than-normal delays. To overcome this, we require each gate-output pair to be tested in at least two crossing paths with only one gate in common. The probability that a delay failure in a gate is masked in both paths would be much smaller than that testing only in one path. The selection process can start with the test set selected in the second criterion. More tests are included if they can help to maximize the coverage according to the third criterion.

We compare the three criteria in terms of several issues:

(1) Number of tests and testing quality - The first criterion produces the largest test set covering a larger number of faults, and the second produces the smallest test set but with smaller fault coverage.

(2) Amount of information to help the later phases of test generation - The second criterion can indicate which gate-output pairs are not covered. This provides valuable information for future test generation. In addition to this, the third criterion also indicates which gate-output pairs in the second criterion cannot be tested in two crossing paths. The first criterion only directly indicates which delay faults are not covered. This information is less useful than that in the form of gate-output pairs, since the number of such faults is usually too large. To overcome this, an algorithm can be added to analyze the resulting coverage in terms of gate-output pairs.

(3) Computing time - The second criterion is the most time-effective. The first criterion could be more time-effective than the third, when the number of gates is large. However, if the post-selection analysis mentioned above is also included, it could be the most time-consuming among the three criteria.

EXPERIMENTAL DATA AND DISCUSSIONS

The experimental test selector has been applied on a 4-bit off-the-shelf ALU chip, Texas Instrument 74S381 (Fig. 6). The ALU circuit has 12 primary inputs, 6 primary outputs, 67 gates (including 4 exclusive-OR gates at primary outputs), 14 inverters and 36 fan-out points. There are totally 296 nodes and 1114 paths. The faults to be considered are 592 single stuck-at faults, 2228 delay faults (two for each path) and 134 gate delay failures (two for each gate). A stuck-at-fault test set with 14 test patterns is shown in Table 1, which was generated by hand to partially check its functions. It is found to be able to cover 455 single stuck-at faults (76.8%) in the circuit, including all input stuck-at faults. Results of the Criterion 1 selector (to cover as many delay faults as possible) show that 319 delay faults (14.3%) can be covered by all delay tests derivable from the stuck-at-fault tests. A near-minimal delay test set is found to be derivable from test patterns 1,2,3,4,5,6,7,8,9,11,12 and 13. It can be applied as 101 single transition delay tests, or equivalently as 51 multiple transition delay tests (with no masking effects). The Criterion 2 selector (for covering as many gate-output pairs as possible) produces a test set covering 116 gate-output pairs (86.6%). The set can be applied by 39 single transition delay tests or 24 multiple transition delay tests, which are based only on test patterns 1,2,3,4,5,6,7,11 and 12. The Criterion 3 selector (for covering as many gate-output pairs in at least two crossing paths as possible) finds that no tests deriable from the stuck-at-fault test set exist which can improve the Criterion 2 test set, due to the two-crossing-path requirement. The test set covers 30 gate-output pairs (22.4%) by two crossing paths and 86 gate-output pairs (64.2%) by only one path.

This experiment shows that a fairly good coverage (for example, 86.6% goal of Criterion 2) can be obtained by a small number of tests (24 multiple transition tests). If we start with a larger stuck-at-fault test set with more than 90% or 95% coverage, the result can be more impressive. The execution time also seems to be feasible for practical circuits. The experiment for 74S381 takes 16 minute CPU time on an Itel AS-6 computer.

We would like to point out our effort was to develop the first phase for delay test generation which generates efficient delay tests. The resulting test set is by no means complete by any criterion, however it is the best that can be derived from a set of given test patterns. The uncovered delay faults or gate-output pairs are reported by the selector program. The tests for these uncovered faults can be generated by the path sensitization method such as that used in reference 2. In case the set of delay faults not covered are prohibitively large, only the important paths among them can be selected for test generation. Such important paths could be a small set of the longest paths, which can be identified

by inspection or engineering judgement (e.g. for adders), or by selecting paths with the largest number of gates.

CONCLUDING REMARKS

In this paper, we have proposed a new approach for test generation of delay faults in logic circuits. We use a stuck-at-fault test set as the starting point, and derive delay tests from it. This scheme has the advantage that multiple input transition tests are generated, which detects several paths at the same time. Since a stuck-at-fault test is used as the second test pattern in a delay test, the application of delay tests will automatically detects stuck-at faults also. Thus this method can be used for manufacturing testing as well as field testing for mixed faults (combination of stuck-at faults and delay faults).

Because solid delay faults produce errors only in some occasions which are difficult to identify, they are likely to be observed as intermittent or transient faults. Sometimes, a delay failure itself could be intermittent. For example, drifting of component delays due to aging can drive the path delays out of the safty margin, and **eventually** present intermittent delay faults. It is suspected that some unidentifiable intermittent or transient faults observed in field operation are actually delay faults. Since intermittent faults account for a substantial fraction of overall field maintenance cost, it seems reasonable to apply delay tests with stuck-at-fault tests in periodical field maintenance. Our approach helps to produce an efficient test set for testing the combination of these two kinds of faults.

REFERENCES

[1] E. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," in Proc. 14th Design Automation Conf., New Orleans, June 20-22, 1977, pp. 462-468.

[2] P. Hsieh, R. A. Rasmussen, L. J. Vidunas and W. T. Davis, "Delay test generation," in Proc. 14th Design Automation Conf., New Orleans, June 20-22, 1977, pp. 486-491.

[3] John B. Peatman, Digital Hardware Design, McGraw-Hill, Inc. 1980, pp. 265-266.

[4] J. D. Lesser and J. J. Shedletsky, "An experimental delay test generator for LSI logic," IEEE Trans. Comput., vol. C-29, pp. 235-248, March 1980.

Test No.	Input Patterns												Numbers of One-Dimensional Sensitizing Paths
	S_2	S_1	S_0	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	C_n	
1	0	0	0	0	1	0	1	1	0	1	0	1	22
2	0	0	1	0	1	0	1	1	0	1	0	0	31
3	0	0	1	1	0	1	0	0	1	0	1	1	33
4	0	1	0	0	1	0	1	1	0	1	0	0	29
5	0	1	0	1	0	1	0	0	1	0	1	1	27
6	0	1	1	1	0	1	0	0	1	0	1	0	58
7	0	1	1	0	1	0	1	1	0	1	0	1	62
8	1	0	0	1	0	1	0	0	1	0	1	0	20
9	1	0	0	1	1	1	1	1	1	1	1	1	20
10	1	0	1	1	0	1	0	0	1	0	1	0	*
11	1	0	1	0	0	0	0	0	0	0	0	1	18
12	1	1	0	1	1	1	1	1	1	1	1	0	28
13	1	1	0	1	0	1	0	0	1	0	1	1	19
14	1	1	1	0	1	0	1	1	0	1	0	0	*

*: not available

Table 1. The starting stuck-at-fault test set used for selection

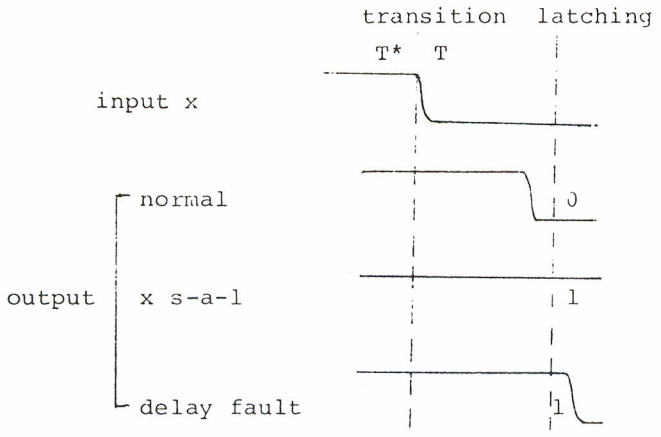


Fig.1 Single path sensitization - assuming even number of inversions on the path.

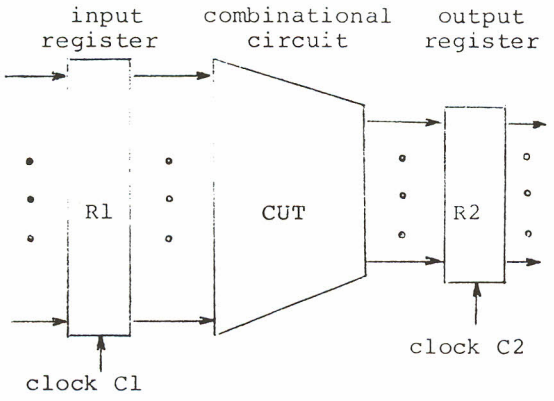
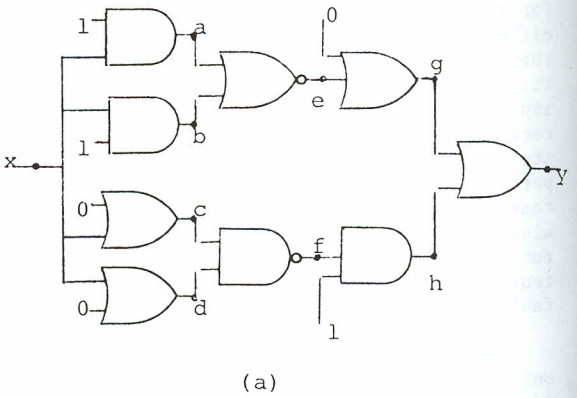
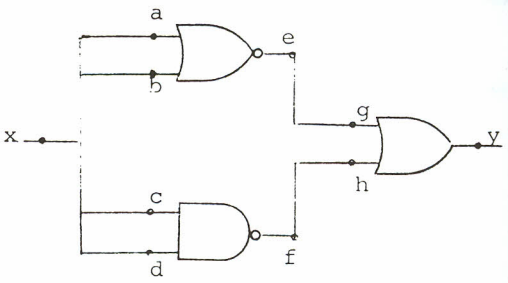


Fig.2 A simplified model for the combinational circuit-under-test (CUT) and its testing environment.

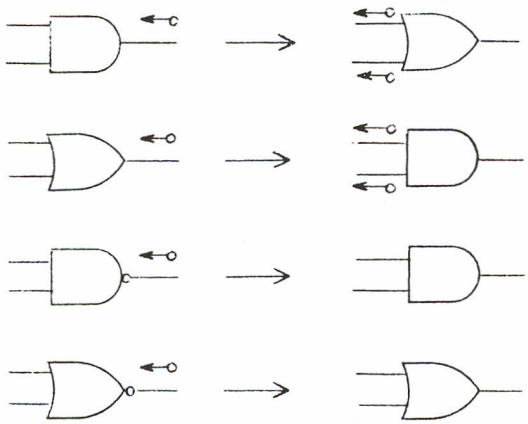


(a)



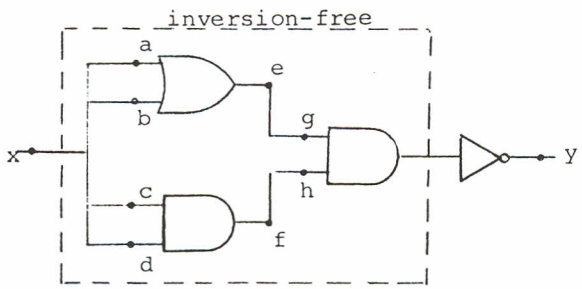
(b)

Fig.3 A sample circuit and an abstracted subnetwork.



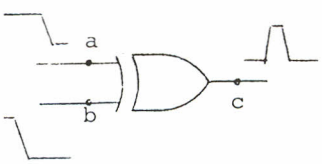
The symbol ◄o indicates a movement of inversion.

(a)

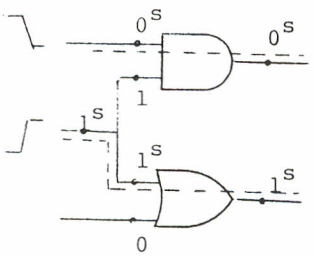


(b)

Fig.4 The transformation rules and the abstracted subnetwork after transformation.



(a)



Superscripts 's' indicate sensitizable nodes.

(b)

Fig.5 Masking effects of multiple input transitions.

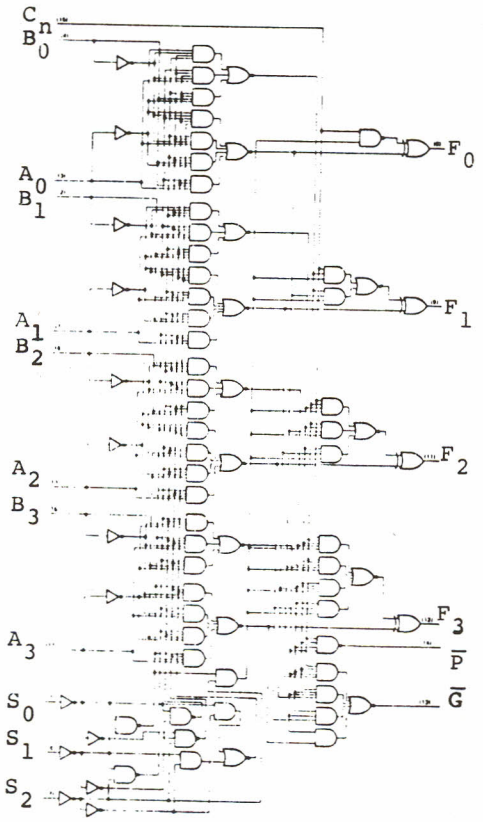


Fig.6 The circuit diagram of Texas Instrument 74S381.