設 計 自 動 化 77－5
(1995. 10. 19)

題目： LSI故障解析支援システム

発表者名： 後藤正樹、小谷浩、白鳥文彦
勤務先： 株式会社日立製作所 デバイス開発センタ
ＤＡ開発部 ＤＡ４グループ
連絡先： 〒１９８東京都青梅市今井２３２６番地
ＴＥＬ：０４２８－３２－１１１１内線５２４４
ＦＡＸ：０４２８－３３－２１５７
E-mail : mgotoh@ddc.hitachi.co.jp

あらまし

LSIの故障位置を指摘するシステム「故障位置指摘システム」を開発した。本システムは、ほとんどすべてをソフトウェアによって構築している。以前のものに較べ、対象となるLSIの規模が４倍になっても、最低でも１０倍の速さを実現した。このシステムは２５０ｋゲート規模の短縮故障を指摘するのに、たかだか数十分しかかからない。また、指摘した故障位置を論理図及びレイアウト図の上に示して見ることができる。さらに、異物検査装置を使って発見された不良箇所と、本システムの結果との関係をレイアウト図の上に示すことができる。

キーワード 故障解析、故障位置指摘、故障ｼﾐｭﾚｰｼｮﾝ、ﾀﾞｲﾚｸﾄｽｷｬﾝ故障ｼﾐｭﾚｰｼｮﾝ、ﾃﾞｨﾚｲ故障

Title: A Fault Location System for Complex LSIs

Authors: Masaki Gotoh, Hiroshi Odani, Fumihiko Shirotori
Affiliation: Device Development Center, Hitachi Ltd.
Design Automation Development Department, DA4 Group
Address: 2326 Imai, Ome-shi, Tokyo 198
Phone: 0428-32-1111 Ext. 5244
Fax: 0428-33-2157
E-mail: mgotoh@ddc.hitachi.co.jp

Abstract

We will present a fault locator that is built almost entirely by software tools. The system speed has increased tenfold compared to its predecessor even though the complexity of the LSI being analyzed has, at minimum, quadrupled. The system can locate a single fault from a 250K gate circuit in a matter of minutes. The fault can then be viewed in a schematic or layout diagram. Furthermore, particles detected from the particle inspection unit can be viewed with the layout diagram to find correlations between these particles and the fault locator results.

key words Fault Analysis, Fault Location, Fault Simulation, Direct Scan Fault Simulation, Delay Fault

An Automatic Fault Location System for Complex Logic LSIs

M. Gotoh H. Odani F. Shirotori

1. Introduction

The list of equipment that are used for fault analysis is staggering – AEM, EPMA, EDX, WDX, ESCA, UPS, SAM, SEM, SIMS, TEM, XD, FIB, EBP, AES. Each type has its own characteristics, it's own ups and downs. Millions of dollars must be spent (initially) to locate a single fault within a device the size of a fingernail. Yet without them, fault analysis would be impossible.

With the introduction of CAD tools, the many phases of LSI design have been sped up, automated, and made more efficient. Fault analysis, however, has not been greatly affected by software tools. The algorithms are slow, the data size is enormous, the results are inaccurate, and the models are unrealistic. On the other hand, the hardware required for fault analysis has improved drastically. In the past, software simply could not be trusted to locate and analyze the faults.

The system developed at Hitachi, the Automatic Fault Location System, has proven to be effective against all the above mentioned problems. The system can quickly locate faults within 100+ k gate LSIs in a matter of minutes using only CAD tools. The located fault is then analyzed using a combination of software and hardware tools. Having been tested with actual data, the accuracy of the system has been confirmed.

Furthermore, the system supports not only DC function tests, but delay tests as well. With LSIs running at speeds of over 100 MHz, DC function tests are not sufficient to ascertain the reliability of the chip. To obtain the maximum performance of the LSI, delay faults must be a factor to consider.

Finally, the system also considers ease-of-use, a factor sometimes neglected in the design of CAD tools. While performance, functionality, and reliability are of the utmost importance, the system must also be simple to operate from the user's point of view. This system supports all the comforts of the X-window system. With easy-to-use mouse-driven commands, schematic and layout diagrams can be opened to confirm the location of the fault. Also, data obtained from the particle inspection unit can be read into the layout diagram to determine the correlation between the possible faults and the detected particles on the wafer. With this information, the user can begin the fault analysis phase.

2. Basic Concepts: A Review

This system is based upon two key components: Fault Simulation and Fault Location. Fault simulation is used to find the fault coverage of the test vectors and to build the fault dictionary. Fault location compares the fault dictionary with the actual pins that failed during testing. A fault candidacy list is created by the fault locator with the possible faults listed in order of probability.

The creation of the fault dictionary is by far the most complicated and time-consuming process of the entire system. However, the algorithm itself is quite simple. Most circuit defects can be observed using a boolean model of the defect (stuck-at model). Therefore, gate-level simulation is sufficient to observe faulty behavior.

First, the simulator maps the logic circuit into a series of boolean truth tables for each gate or cell to be simulated. Hereafter, these gates or cells that make up the basic building blocks of the circuit will be called *logic primitives*. The simulator then feeds the test vectors into the input pins. The behavior of this circuit is called the *"good machine"* (figure 1). The logic values traverse the circuit and the results are obtained from the output pins. Next, for every possible fault point, or *fault origin*, the simulator inserts the fault into the circuit and simulates. Finally, the simulator compares the behavior of this *faulty machine* with that of the good machine (figure 2). If the differences in behavior can be observed at the output edge pins, the fault is said to be detected.

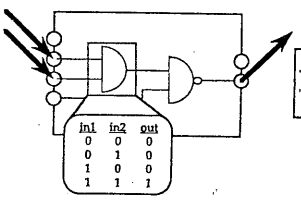

figure 1. good machine simulation

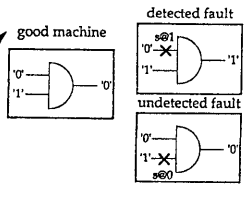

figure 2. fault simulation

Often times, the faulty machines are simulated in parallel to obtain greater speedup. By using a hardware fault simulator, not only does each simulation complete faster, a concurrent simulation model not possible with software simulators can be implemented to further enhance speed. The concurrent model simulates only those portions of the circuit affected by the inserted fault. For example, if a fault is inserted into the last gate of a chain of gates, only the last gate need be resimulated (figure 3). Even greater speedup can be obtained by combining the parallel and concurrent fault simulation models.

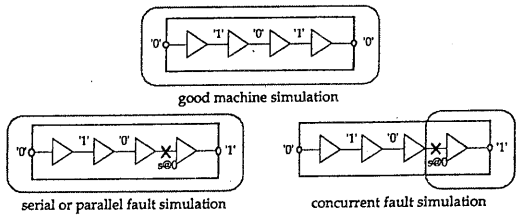

figure 3. fault simulation methods. The gray line represents the area to be simulated.

	tester results	dictionary		fault locator			
		fault	detected pin	# of failed pin	faults	detected	contradict
time 100	pin 1	fault A	pin 1,2,3		A	3	0
	pin 2	fault B	pin 1,3	3	B	2	0
	pin 3	fault C	pin 1,2,3		C	3	0
time 200	pin 2	fault A	pin 2,3		A	2	0
	pin 3	fault B	pin 3	2	B	1	0
		fault C	pin 1,2,3		C	2	1
time 300	pin 1	fault A	pin 1		A	1	0
		fault B	pin 1	1	B	1	0
					C	0	0

figure 4. fault location. Note the contradictory value for test 2, fault C.

The *fault dictionary* contains the output of the fault simulation. For every fault, there is a record of the fault and its status (detected or undetected). If detected, the dictionary shall contain the time of detection and where it was detected (i.e., which output pin failed). These records are sorted in ascending order by time (figure 4).

The fault locator compares the actual failed pins and the fault dictionary. In figure 4, the test results show that pins 1, 2, and 3 failed during the first three tests. The second table contains the data in the fault dictionary. The third table lists the results of the comparison. In test 1, faults A and C can be detected from the three failed pins. In test 2, faults A and C can be detected from the two failed pins. However, if fault C is the actual fault, pin 1 should also fail. Since it did not, this fault is said to contradict the actual results. Finally, test 3 is compared. The fault candidacy list summarizes the results (figure 5). Fault A is placed in the level 1 category. Level 1 indicates that the faults in this set are the most probable faults. Both faults B and C are placed in level 2. The contradicting value in test 2 causes fault C to be set equivalent to fault B even though it had been detected more times than fault B.

Total number of failed pins: 6

Level	Fault	Detected	Contradictions
1	A	6	0
2	B	4	0
	C	5	1

figure 5. Fault locator results. In this case, fault A would be the most likely fault. Faults B and C, being in the same set, are equally likely to be the actual fault.

These two procedures make up the core of the system. Several other factors effect both fault simulation and fault location such as fault collapsing and observability/controllability. These issues will not be discussed in this paper.

3. System Overview

Figure 6 illustrates the basic system organization.

The system is divided into five phases. The first two phases are executed after the designs are completed but before manufacturing. The final three phases are executed after an LSI fails during testing.

(a) Conversion Phase

This phase converts the different formats for test patterns, netlists, and layout patterns to the system's predefined format (ZyCaD Input Record, EDIF 200, and GDSII respectively).

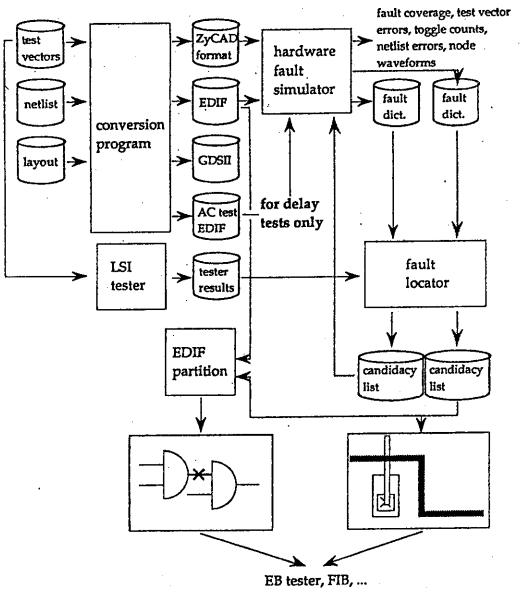

figure 6. A simplified system flow diagram. The gray line represents the $n = \infty$ simulation/fault location.

(b) Simulation Phase

The fault simulator creates the fault dictionary using ZyCaD Corporation's Paradigm XP-2016 Hardware Fault Simulator. During this phase, faults are collapsed while faults impossible to detect are removed from the fault list and marked as such. The simulator also supports three other states that a fault can exist in other than detected, undetected, and impossible – possibly detected, hyperactive, and oscillatory. These three types of faults are considered to be undetected by the system. The simulator is initially set such that if a fault is detected, it is immediately deleted from the list (fault dropping or n=1 run). Therefore, every fault can be located at most once. When the simulation is run a second time using the reduced fault origin list (as explained in the next phase), detected faults are never removed from the list (full fault simulation or n=∞ run). Therefore, every fault can be located as many times as the test vectors allow. This double simulation process was chosen to decrease simulation time.

Some programs that do not directly pertain to simulation are also run during this phase. These include the delay model netlist build, and the schematic diagram database used for partitioning the schematic diagram. These programs are run here because they can be executed in parallel with the simulation. Since fault simulation requires the most time, this would be an ideal time to run other programs that are not dependent on the simulation results.

Also, this phase is used to evaluate data for uses other than fault analysis. These include fault coverage calculations, toggle checking, design error checking, and test vector error checking.

(c) Locator Phase

Here the fault dictionary is compared with the tester results using the algorithm described above. As mentioned in the previous section, this phase is run twice, once during the fault dropped simulation and once during the full fault simulation. After the first run, the fault candidacy list is greatly reduced, often to the order of thousands or tens of thousands. Since the faults are removed from the list immediately after detection, every fault will be a member of the Level 1 set (with the number of detections equal to 1). These faults are reset as undetected and rerun using full fault simulation. This time, however, since the faults are not removed after detection, the number of times the fault was detected will also be recorded in the fault dictionary. Therefore, a more accurate fault candidacy list can be built.

(d) Isolation Phase

During this phase, user interaction becomes most important. The programs themselves simply display the layout and schematic diagrams.

Although the entire layout diagram can be shown, the schematic diagram must be partitioned if too many primitives exist in the circuit. A user-definable number of primitives before and after the possible faults are displayed. The fault candidacy list is also read into the diagrams. Furthermore, the particle database is matched with the layout diagram to represent correlations between the particles detected during manufacturing and the results of the fault locator.

(e) Analysis Phase

This phase is technically not part of this system. This is where the actual LSI analysis begins using the above mentioned tools such as FIB, SEM, and so forth. Currently, only the EB tester is directly connected to this system. During simulation, nodes to be probed are monitored and its waveform is output. This waveform is then compared with the waveform probed with the EB tester to isolate the fault. This procedure is executed when the fault locator can not isolate the fault to a single point.

4. Improving Performance and Functionality

Several measures have been implemented to improve performance and functionality in this system. The first three features helped to gain speedup while the last two features provided new functions to assist in the analysis phase.

(a) Reduction of Fault Origins and Gates

The time required for fault simulation is directly proportional to the number of faults, the number of gates, and the length of the test vector. The length of the test vector is a constant (without reducing the fault coverage). The number of faults and gates, however, can be reduced using several common methods such as collapsing, and memory primitive reductions.

(b) Reduction of the Length of Test Vectors

The format of the test vectors used by Hitachi could be adjusted to decrease the test vector count without reducing the fault coverage. For DC function tests, the test vectors created by the automatic test generation system is divided into two "cycles" – 0-cycle and 1-cycle (figure 7). The 0-cycle portion tests the scan circuitry. More specifically, the test vectors pass through the scan circuitry into the scan latches. The contents of the latches are then scanned out without passing through the main logic circuits. The 1-cycle portion tests the main logic circuits using scan. The signals are scanned in, as above, but then go through the main logic circuits and are clocked into the scan out latches. The data in the output latches are then scanned out. While the 0-cycle tests have test vectors in the order of $10^4 \sim 10^5$, the 1-cycle tests contain $10^6 \sim 10^8$ vectors. Furthermore, since the simulator uses a concurrent fault algorithm (only passes that differ with the good machine are simulated), the number of events, or primitive simulations, increases since the 0-cycle test simulates only the scan circuitry while the 1-cycle test simulates the entire chip. The reason for the increase in test vectors for the 1-cycle test is that for each "test", the number of test vectors is at least the number of scan latches (10^3 +). Each "test" scans in data for the input scan latches and scans out data from the output scan latches one test vector at a time while the actual "test" needs only one test vector. In other words, each "test" requires at least 10^3 test vectors. Therefore, the number of "tests" in the 1-cycle test is actually only $10^{(6-3=3)} \sim 10^{(8-3=5)}$.

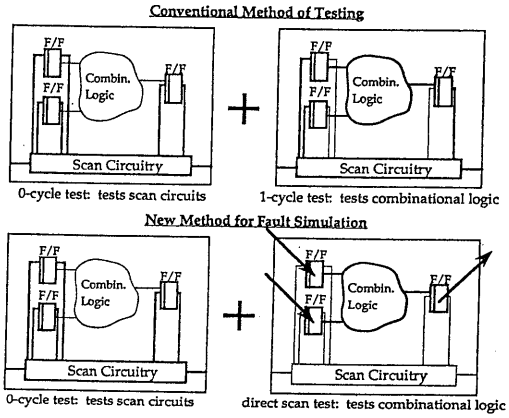

figure 7. Direct scan simulation. By feeding the test vectors directly into the scan latches, the number of test vectors and events decrease dramatically.

The method we use, called the direct scan approach, inserts and extracts data directly into the scan latches simultaneously. In one test vector, the vectors are inserted into the input latches, passes through the main circuitry, and extracted from the output latches. Since the 0-cycle test simulates the scan circuitry, there is no need to resimulate during the 1-cycle test. Furthermore, the number of events decreases since now the simulator tests only the main circuitry and not the entire circuit as before. The speedup gained using this method is proportional to the number of scan latches used.

One problem with this method appears when a single scan latch is used for both input and output. Since data is directly scanned in, clocked, and sensed out during the same test, the scanned in data may fight with the incoming output data that is to be sensed out. Therefore, the functionality of the scan latch must be carefully understood before deciding exactly where to scan in the data (data pin, set/reset pin, inside latch loop, and so forth).

(c) Delay Faults

The ZyCaD hardware simulator simulates only stuck-at faults. However, with LSI speeds easily exceeding 100 MHz, delay faults must be another factor to consider. However, no hardware simulator currently exists that supports delay faults. Furthermore, software simulators are too slow for large circuits. Therefore, a method must be introduced that can convert stuck-at faults to delay faults.

First, a simple introduction of delay faults is in order. There are two types of delay faults – *setup test faults* and *hold test faults* (figure 8). Setup faults are caused by delayed data into the output latch. Signals that should have been clocked in are not. Hold faults are caused by a delay in the clock pulse. The effective result is that signals that should not be clocked in, are.

If one looks closely at the behavior of setup time faults, it is clear that these faults are equivalent to stuck-at faults during a certain period of time. Therefore, using the delay test vectors in a stuck-at fault simulator, the results can be used for fault analysis. However, hold time faults are not equivalent to stuck-at faults. Sticking a clock node will cause more than just a delayed signal. Therefore, a second method must be considered.

A "delay cell" can be inserted where every fault origin is located. This delay cell will fool the simulator into testing for delay faults while its algorithm is simulating for stuck-at faults. Figure 9 represents the delay cell. The switch A is a 2-to-1 selector that chooses between the normal path and the buffered

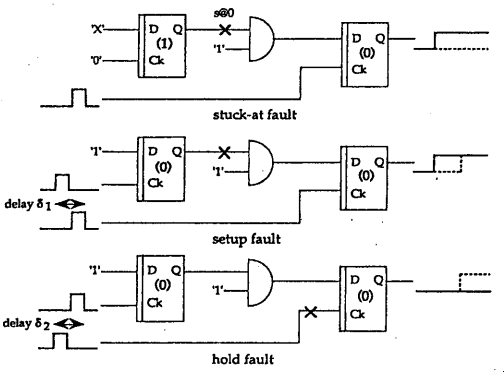

figure 8. Types of delay faults. The paranthesized values represent intial values in the latch. The dashed line represents the signal when a fault occurs.

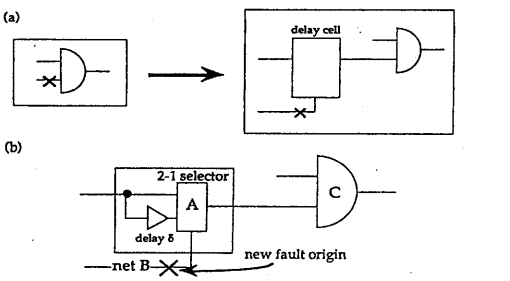

figure 9. Delay fault support. (a) A delay fault can be converted to a stuck-at fault by inserting the delay cell where the delay fault is to be simulated. (b) The delay fault itself is a delay primitive and a switch.

path. The buffer is a delay primitive with some delay δ that will delay the signal if chosen by the selector. The net B selects which path to take. Normally, net B is held at logic-0. When simulating a stuck-at-1 on this net, the signal will be delayed and a delay fault will occur. Therefore, a delay fault at the input of gate C is equivalent to a stuck-at-1 fault on the selector net.

(d) Particle Database

When viewing the layout diagram, the faults specified by the fault locator will be visible. In addition to these faults, the particles detected during particle inspection can also be viewed on top of the faults and the layout diagram. In this way, the user can examine if a foreign particle is the cause of the fault even before physically checking the circuit.

5. Results

circuit	gates (FF)	fault origins	test	test vectors	logic	fault (n=1)	fault coverage
A	108K (2.1K)	85K	0-cycle	60K	44s	35m27s	28.9%
			1-cycle*	3,486K	36m05s	49h34m49s	76.6%
			1-cycle	1.5K	59s	37m08s	54.8%
B	250K (4.1K)	255K	0-cycle	94K	1m37s	1h33m57s	18.1%
			1-cycle*	14,183K	3h41m40s	***	***
			1-cycle	4.0K	6m01s	5h54m30s	78.5%
C	208K (4.3K)	151K	0-cycle	95K	1m37s	1h27m15s	31.5%
			1-cycle	5,304K	1h24m45s	***	***
			1-cycle	1.4K	2m17s	1h35m39s	56.8%
D	169K (3.3K)	120K	0-cycle	85K	1m10s	37m20s	28.2%
			1-cycle	1.7K(5,925K)	2m05s	49m12s	63.7%
E	172K (3.5K)	128K	0-cycle	77K	48s	34m54s	32.9%
			1-cycle	3.2K(6,636K)	3m40s	1h45m10s	60.2%
F	255K (5.6K)	178K	0-cycle	108K	2m02s	1h45m13s	33.7%
			1-cycle	4.2K(15,244K)	9m11s	4h32m11s	56.3%

note:
1) "Logic" refers to the good machine simulation time.
2) "Gates" refers to the number of ZyCaD primitives and not the actual number. However, FF refers to the actual number of scan latches within the circuit.
3) Both the conventional 1-cycle test and direct scan 1-cycle test were run for circuits A, B, and C. However, the conventional 1-cycle fault simulation was run for circuit A only since the time to run it for other circuits was too long.
4) The fault coverages represent the ratio of detectable faults for that test only. During actual testing, a function test is run in addition to the 0- and 1-cycle tests for a total fault coverage of 99.99%.

table 1. Logic and fault simulation times for actual circuits developed at Hitachi.

Table 1 represents simulation times for several circuits. One can see that fault simulation takes up most of the time required by the system. Since this phase is executed only once and before a failed circuit is found, the actual time required once a fault is located is the sum of the execution times of the fault dropped fault location, the full fault simulation and location, and the diagram display. Also note the speedup attained from using the direct scan approach for the 1-cycle tests during fault dropped simulation.

sample	fault location (n=1)	# of faults (n=1)	fault sim (n=∞)	fault location (n=∞)	# of faults (n=∞)
1	37m41s	142	5m54s	57s	2
2	33m11s	286	5m55s	50s	1
3	36m13s	62	15m32s	3m15s	3
4	1h08m22s	493	20m44s	7m03s	2
5	1h07m06s	740	11m35s	5m54s	1
6	47m14s	65	10m22s	26s	2
7	36m11s	17	11m35s	25s	6
8	32m40s	92	10m24s	21s	1
9	31m12s	27	6m56s	19s	1
10	30m48s	25	10m27s	16s	1
11	31m15s	122	11m06s	1m09s	4
12	30m48s	51	10m24s	21s	1
13	35m01s	111	32m41s	22m13s	4
14	52m22s	39	11m20s	27s	1
15	58m49s	215	19m04s	3m31s	1

note:
1) The execution times for fault location is dependent upon the CPU usage ratio of the host. These values were obtaining using 30% of the CPU resources.
2) "# of faults" refers to the number of Level 1 faults (the total number of faults is equal to the Level 1 faults for the n=1 fault location).

table 2. Fault location execution times for several samples of Circuit B from table

Table 2 shows the execution times for the fault dropped fault location, full fault simulation, and full fault location (the time for diagram display is negligible). One can see that, except for a few cases, a single fault origin could be isolated even before viewing the circuit.

6. Conclusion

Although the fault dictionary approach has been criticized as being too time-

and resource-consuming, the fault locator system has proven to be successful in identifying faults very quickly. Although the system by itself can only speculate the location of the faults, the system can speedup the analysis phase of fault analysis by giving the user a highly probable location of the fault. Furthermore, the system can simulate delay faults, a feature that is lacking in many conventional fault analysis systems. The system has already provided much speedup toward the completion of the next generation mainframes and workstations by Hitachi.

7. Acknowledgments

The authors would like to gratefully acknowledge the following people for their help and support in their work:

Kuniaki Kishida, Hitachi Computer Engineering, Inc.
Tomoji Nakamura, Hitachi Computer Engineering, Inc.
Masato Hamamoto, Hitachi Ltd., Device Develop. Center, First Design Dept.
Yoshiaki Anata, Hitachi Ltd., Device Develop. Center, Second Design Dept.
Masanari Ishii, Hitachi Ltd., Device Develop. Center, DA Develop. Dept.

8. References

[1] P.L. Crook, "Evolution of VLSI Reliability Engineering," Proc. Int'l Reliability Physics Symposium (IRPS), 1984

[2] K. Nikawa, "VLSI Failure Analysis: A Review," Microelectron. Reliab., Pergamon Press. Vol. 32, No. 11, 1992.

[3] H. Cox and J. Rajski, "A Method of Fault Analysis for Test Generation and Fault Diagnosis," IEEE Trans. on CAD, Vol. 7, No. 7, 1988.

[4] J. Kato et. al., " Fault Diagnosis Based on Post-test Fault Dictionary Generation," Proc. Int. Test Conf., 1989.

[5] E.B. Eichelberger and T.W. Williams, " A logic design structure for LSI testability," Proc. 14th DA Conf., 1977.

[6] C. C. Liaw et. al., "Test Generation for Delay Faults Using Stuck-at-Fault Test Set," Proc. Int'l Test Conf., 1980.

[7] T. Hayashi, et. al., "A delay test generator for Logic LSIs," IEEE FTCS-14., 1984.