Modeling Contemporary Faults

Ravishankar Murugesan, Hemant Savla

***Abstract*—A survey paper by Abraham and Fuchs published in 1986 provided a reference for fault models. Fault modeling is still an active area of research, but to our best knowledge, there is no paper that surveys research and trends in this field. This paper describes models for these contemporary faults: stuck-open and stuck-on CMOS faults, delay faults, bridging faults, and crosstalk faults. Apart from surveying fault models, we present simulation results of models that were either developed by us or others, along with some recommendations.**

***Index Terms*—Fault models, contemporary faults, transistor faults, delay faults, bridging faults, crosstalk faults.**

I. INTRODUCTION

The stuck-at fault model is perhaps the most widely used model for faults in digital circuits. Although many faults occur at the transistor-level, the stuck-at model is applied at the gate-level. In spite of this apparent mismatch, the stuck-at model has proven to be the de facto standard for fault modeling. As VLSI technology keeps deepening, the relevance of this model is brought into question every now and then. Along with that, new models and testing techniques were proposed to test faults that were previously ignored. This paper surveys important research in contemporary faults, with particular focus on modeling those faults. By using the term contemporary, we restrict our work to these specific faults: stuck-open and stuck-on transistor faults (hereafter referred to as general CMOS faults), bridging faults, and crosstalk faults. These faults are important in view of their relevance to today's level of IC integration. We produced this paper keeping safety-critical systems in mind; for in such systems, detailed fault modeling and simulation is necessary.

It is interesting to recognize that present industry focus is still on testing faults using stuck-at models, so research that aims to develop gate-level abstractions of faults are important. Although the stuck-at model does not directly represent many physical failures in circuits, it has two strong arguments in its favor: (1) all test generation and fault simulation algorithms were originally developed for this fault, as it can be thought of as technology independent and well-suited for logic simulation, and (2) many faults can be deduced as stuck-at faults, by observing their behavior. The validity of the stuck-at model is brought into question time and again, and there are proponents who call for other testing methods, like circuit simulation [1], to test faults in CMOS circuits. However, circuit simulation is computationally expensive and time consuming, and further disadvantaged by the fact that the complexity of ICs keeps increasing. Stuck-at fault test generators are very mature, so, if faults can be reliably modeled to be processed by such test generators, which would save a lot of time and effort. Part of our work concentrates on such research.

R. Murugesan and H. Savla are with the Department of Electrical and Computer Engineering at the University of Wisconsin – Madison, Madison, WI 53706.

II. GENERAL CMOS FAULTS

The stuck-at model can be reliably used in bipolar and nMOS circuits, since it represents a variety of physical failures. However, in CMOS circuits, the process irregularities that result in shorts and opens cannot be represented directly as stuck-at lines. These faults must certainly be tested for; otherwise unrealistic fault coverages would be obtained. In this section, stuck-open and stuck-on faults are described. Two other types of faults that occur at the transistor level, namely delay faults and bridging faults are dealt with in separate sections. The relevance of the stuck-at model to general CMOS faults was investigated in [2], so this paper will not address that question in detail. However, some papers that were published after [2] dealt with this, and we summarize those results in this section.

2.1. Stuck-open faults

When a FET is locked in a non-conductive state independent of the control signal, a stuck-open fault results. Such cases include missing source and/or drain contacts, and broken connection lines. Gate-source shorts are also treated as stuck-opens. In the presence of this fault, a high impedance node may result, which retains its previous state. The detection requires at least two vectors, due to the sequential behavior. The first vector initializes the gate output to a logic value opposite to that driven by the network, and the second activates the fault by creating a conducting path between one of the power supplies and the output of the fault-free gate. This vector must also propagate the gate output to primary outputs.

2.2. Stuck-on faults

When a FET is locked in a conductive state independent of the control signal, a stuck-on fault occurs. These represent physical failures such as drain-source shorts and errors in threshold voltage adjustment. In the presence of such a fault, a signal node can be connected to both power supply and

ground, and it is driven to an intermediate voltage. It is worthwhile to note that a large percentage of these faults are undetectable.

2.3. Early research

The problem of applying stuck-at tests to detect stuck-open and stuck-on faults, was studied in the early 80s, and the recommendations of such research were summarized in [3]. In one case, it involves adding extra circuitry to each CMOS gate to model the transistor faults as stuck-ats [4, 5]. This is clearly infeasible in large VLSI circuits. The other cases, which required transformations to derive gate-level equivalents of FCMOS gates [6, 7], had significant drawbacks. In a study carried out by [8], it was found that many defects in a typical circuit were undetectable by a logic-level test.

2.4. A Fault Modeling Procedure

In [9], a fault modeling procedure was described to create a database with gate level faults. From the point of view of our survey, this method seems relevant and promising. New fault models were created to increase the accuracy of fault modeling. They are (1) Conditional faults: This is a fault that can be detected only if one of more conditions are satisfied, and (2) Function Conversion: This occurs when a Boolean function g is converted into a gate with the Boolean function f by a physical defect.

2.4.1. General Method: Starting from a switch level description of a standard cell library, an appropriate fault model is chosen to model realistic faults. Then, for each cell in the library, the effect of each fault on that fault model is determined – this is done using analog simulations. The observed fault effects are characterized through gate-level faults. Based on this analysis, a database is created that can be used to generate a gate-level fault list for a given circuit. By this process, effective test patterns are generated based on low level fault model, and using gate-level tools.

2.4.2. Simulation Results: To support the results made in [9], simulation was carried out for a FCMOS OR gate and a complex FCMOS gate implementing the function (D + A (B + C))'. A portion of the simulation results are shown below. For the OR gate, the same results were obtained as in [2]. In Figure 2, it is seen that a short of A and line 1 can be modeled as function conversion of OR -> NOR, with input B stuck-at 0. Others, like B shorted with 1 and 2 shorted with ground manifest themselves as the primary output stuck-at 1. For the simulation of the circuit in Figure 3, similar results were observed. In this circuit, more cases were observed where the fault effect due to a short could not be observed as the manifestation of a stuck-at fault. This further proves the reason to use comprehensive fault models.

2.4.3 Results for Static CMOS: In [9], the influences of all shorts and opens were determined for gates of the libraries. Table I summarizes the results for intra-gate shorts and opens in the static CMOS library. The table clearly shows the need to use non-classical fault models to reflect complex fault effects. For systems with safety critical applications, such comprehensive testing is necessary to assure quality of service.

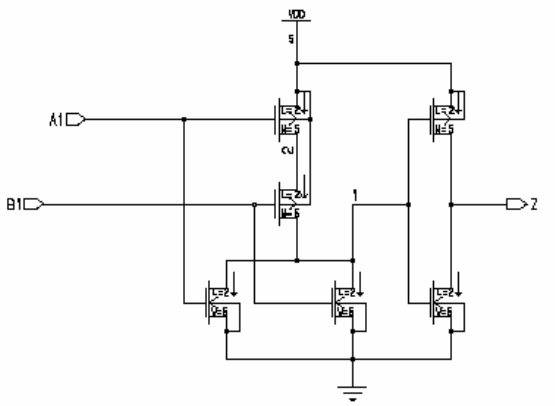

Fig. 1. CMOS OR gate. Lines 1 and 2 denote lines that were shorted in different combinations with the inputs.

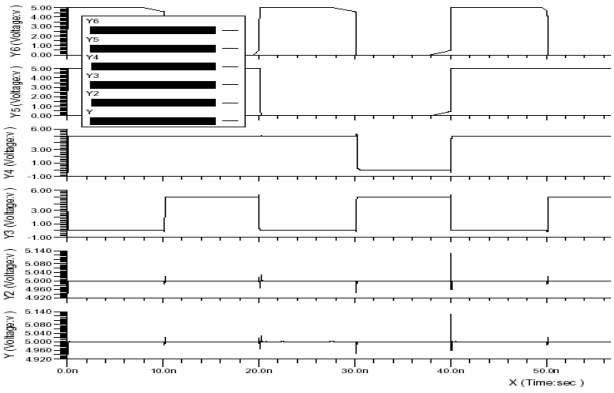

Fig. 2. Some simulation results of CMOS OR gate. The top two waveforms denote inputs A and B. The next waveform denotes the correct output. The next denotes the output when A is shorted with 1. The next two are the outputs (Z stuck-at 1) when 2 is shorted with the ground, and when B is shorted with 2.

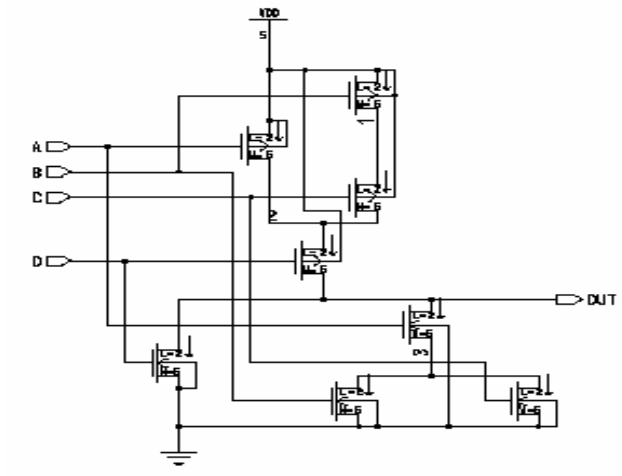

Fig. 3. Complex gate (D + A (B + C))'. Lines 1, 2, and 3 denote lines that were shorted in different combinations with the inputs.

TABLE I
RESULTS FOR STATIC CMOS

Fault Model	Coverage (%)		
	Shorts	Opens	Total
Stuck-at	78.3	-	46.2
Transition	-	66.7	26.5
Conditional transition	-	33.3	13.2
Function	5.8	-	2.6
Conditional function	11.6	-	6.8
Bridging "unknown"	4.3	-	3.4

2.5. Test Vector Generation

An interesting result was reported in [10]. This is presented in Figures 4 and 5. They found that for stuck-open faults, test sequences generated in pseudorandom fashion give a better indirect coverage than deterministic ones. This is because the former is longer, and hence has a greater probability of matching the condition required for stuck-open fault detection, that is, the need for at least two test vectors to detect a fault.

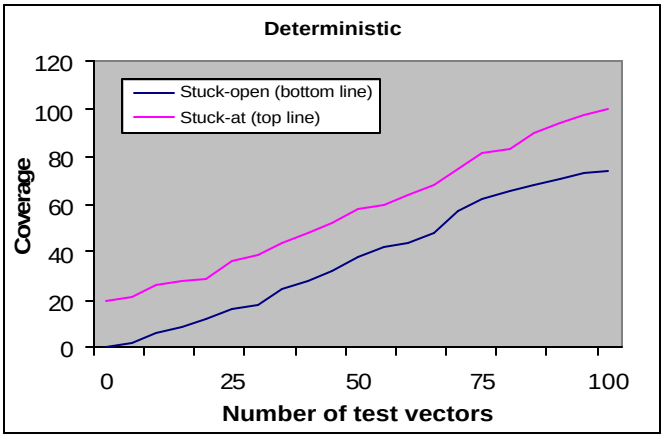

Fig. 4. Coverage with deterministic vectors

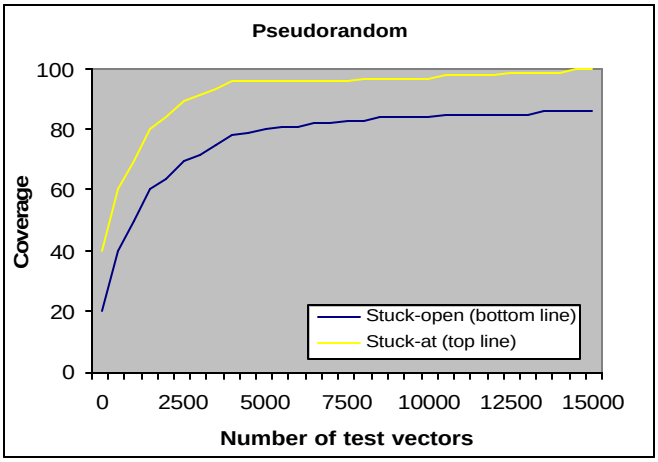

Fig. 5. Coverage with pseudorandom vectors

III. DELAY FAULT

In logic circuits, failures that cause them to malfunction at the desired clock rate are modeled as delay faults. This category of faults has received a lot of attention lately, due to increasing circuit operation frequencies. This section will present an overview of existing delay fault models, and will review trends and recent research that gives an indication of which fault model to use.

3.1. Delay Fault Testing

The basic hardware model for delay fault testing was presented in [11], and it is reproduced in Figure 6.

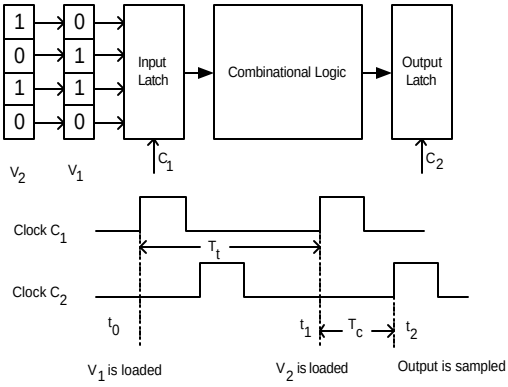

Fig. 6. Hardware model with clock timings

For any circuit, the maximum clock rate is determined by the propagation delays of the combinational logic between latches. During testing for delay faults, two test clocks C1 and C2 are used. The period of test clocks Tt is longer than Tc (the system clock). The activation of the output clock C2 must lag the activation of the input clock C1 by at least DPmax time units (the maximum combinational delay). It is important to note that delay faults do not alter the logic function. Test for stuck-at faults are therefore inadequate for detecting delay faults. Two-pattern test vectors are needed for detection. In the figure above, V1 is the initialization vector, applied at time t0. V2, the propagation vector is applied at time t1. The outputs are sampled at time t2, and (t2-t1) is the rated clock interval Tc. It is almost impossible to exhaustively test for delay faults using the simple method outlined above, as for a circuit with n inputs, the total number of patterns required is 2n(2n-1). So, fault models need to be used for testing delay faults.

3.2. Delay Fault Models

There are five important delay fault models: transition, gate delay, path delay, line delay, and segment. Each of these models is described briefly in this section.

3.2.1. Transition Fault Model: This fault model [12, 13] models a defect that causes a delay in the rising of falling transitions at the inputs and outputs of gates. There are two types of transition faults – slow-to-rise and slow-to-fall. However, this model assumes the presence of a large gate delay defect, which is not a reasonable assumption in today's technologies [14].

3.2.2. Gate Delay Fault Model: This model assumes that delays through logic gates are known with some precision [15]. An important consideration in using this model is to specify the size of the gate delay fault detected by a test T. Much investigation has been done to find the minimum fault size at a fault site [16] – [18], so that the test T at the fault site is guaranteed to detect any fault at that site with a magnitude greater than the minimum fault size.

3.2.3. Path Delay Fault Model: This was first proposed by Smith in [19]. In this model, any path with a total delay exceeding the system clock interval is said to have a path delay fault. This model is proven to be highly effective in covering delay faults, but has a serious limitation because the number of paths to be tested is inordinately large for most practical circuits. There have been research efforts to determine a subset of the total number of paths to be tested [20] – [24], while maintaining the accuracy of fault modeling at this level.

3.2.4. Line Delay Fault Model: This model was proposed to overcome the limitation associated with the path delay fault model. The assumption here is that the delays of all gates are not reduced below their nominal values. The number of faults in the circuit is limited to the twice the number of lines, and the longest structural paths are targeted. This model overcomes the drawback of transition and gate delay fault models; the latter do not model the distributed delay defects along the target path.

3.2.5. Segment Delay Fault Model: This model, proposed in [25], seeks to combine the advantages and remove the limitations of classical delay fault models. The length of the segment L can be as small as 1, or as large as the maximum logic depth. In the former case, this fault model would reduce to transition fault model, and in the latter, it would expand to the path delay fault model. But, like its counterparts, namely transition, gate, and line delay fault models, its effectiveness is not proven.

TABLE II (REPRODUCED FROM [11])
COMPARISON OF DIFFERENT FAULT MODELS

FAULT MODELS	ADVANTAGES	LIMITATIONS
Gate Delay	All gates can be modeled	Distributed failures not considered Exact defect size not possible
Transition	Easy to model all gates	Distributed failures not considered
Path Delay	Distributed failures are considered	Impossible to enumerate all possible paths
Line Delay	All gates are modeled Distributed failures are considered Better coverage metric	Existence of non-robust test May fail for some shorter paths
Segment Delay	Considers general delay defect from spot to distributed defect	Longest delay path may not be tested

This section described delay faults in combinational circuits. For sequential circuits, the considerations are similar, and they are described in [26].

3.3. Effects of Scaling on Circuit Delays

Figure 7 shows the effects of scaling on circuit delays. This is reproduced from Page 50 of [14].

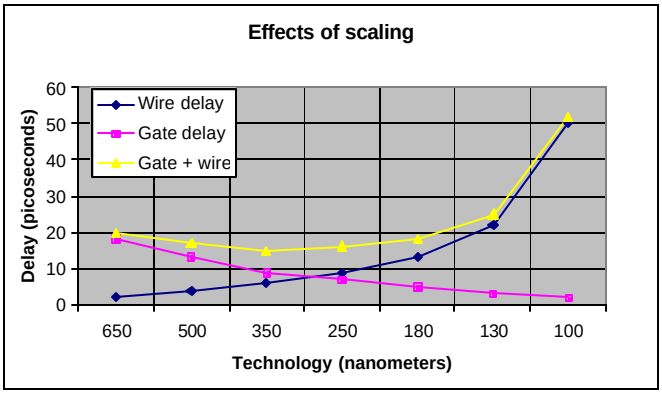

Fig. 7. Effects on scaling on circuit delays

This clearly shows the need for path delay testing, since the wire delay dominates gate delays with the progress of technology. Still, there is a big inherent problem associated with path delay testing, that is, the number of paths can be innumerable, and it is difficult to assign relative importance to them.

3.4. More on Path Delay Fault Model

In spite of the basic problem of this approach, this is the ideal delay fault model. Since its introduction [19], there have been many research initiatives aimed at reducing the complexity of the problem. Some of them are [20] – [24], [27] – [30]. In this section, two methods are outlined, that seem to show significant potential in path delay testing. Since they are not directly related to fault modeling, they are only discussed briefly.

3.4.1. Adaptive Path Selection: Traditional path delay test methodologies use path selection criteria [20] – [24], [29], [30]. While selecting paths, generally, long paths are selected and short paths are ignored. It is possible that defects appear on the short paths, and they escape detection. To overcome this, a new approach called adaptive path selection was proposed in [31], where the test process is divided into two stages, of which the first stage is testing using traditional selection criteria. The idea is to synthesize a new set of paths to be tested using the information of late signals from the first test stage. This selection method makes the path delay test more complete.

3.4.2. Stuck-at test generation: In [32], a test generation method for non-robust path delay faults was proposed, which used stuck-at fault test generation algorithms. As such algorithms are highly evolved; using them to test path delay faults would be very effective. Although this was investigated before [23], [33], there were significant disadvantages, which were overcome in [32]. The given combinational circuit is

transformed into a circuit called a partial leaf dag, after which stuck-at fault test generation is applied to it. Each vector is transformed into a vector pair as a non-robust two-pattern test for the corresponding path delay fault in the original circuit. The complexity of the circuit transformation described depends on the target paths for test generation instead of the total number of paths. The final result is a compact test set with complete fault efficiency, generated in lesser time than a commercial test generation tool.

IV. CROSSTALK / COUPLING FAULTS

Increasing densities of an integrated circuits has led to the interconnect lines carrying the logic signal placed physically close to each other, increasing the likelihood that logic signal and signal changes on one line can affect the logic values on other lines. This phenomenon of introducing noise on one wire by a signal switching on a neighboring wire is called coupling or crosstalk. It is mainly caused by the coupling capacitance between the neighboring wires, although other factors like mutual inductances and substrate coupling also contribute to this kind of transient fault. The four common transient effects of the crosstalk fault as shown in Figure 8 are Positive Glitch (Gp), Negative Glitch (Gn), Rising edge delay (Dr) and falling edge Delay (Dr).

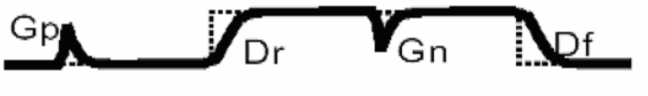

Fig. 8. Common types of the Crosstalk responses

In static logic, crosstalk faults cause unpredictable delay behavior. When neighboring wires are switching in opposite directions, delay increases. However, when signals are switching in the same direction, the delay decreases [34]-[36]. In dynamic logic, it can cause erroneous discharge of stored precharged values, which results in logic errors [40]. In sequential circuit, the cross talk pulses tends to be considered as harmless for synchronous sequential circuits, because generated crosstalk pulses on data lines can be invalidated with enough clock margins, even if the pulse level is high enough to exceed the threshold level. In other words, circuits seem to operate without any problem if all flip-flops accept steady data signal by a slow clock. However, since sequential circuits include not only data line but also clock lines and reset line, the cross talk pulses caused on clock lines and reset lines can lead circuit to erroneous operation if the pulse level is high [38]-[41].

In sub-micron technology distance between lines has decreased and so coupling capacitance and mutual inductance between lines will be large. Also rise / fall times keep getting faster which causes dv/dt and di/dt values to be large. Due to these changes, crosstalk noise between the adjacent interconnects has become an important concern, and if not taken care of, it could lead to malfunctioning of the circuit. Hence, it is becoming increasingly important to model and simulate these crosstalk faults while designing high performance integrated circuits.

First, a few models developed for the analysis of crosstalk between interconnect lines were based on interconnects modeled as transmission lines and used the multi-conductor transmission line theory to analyze crosstalk phenomenon [42]-[44]. Although these models were quite effective for the specific cases, they weren't quite applicable to VLSI circuits because of the complexity of the circuit model and high computation time. Rubio and Kinoshita [45][46] proposed a simplified lumped RC model for crosstalk between a pair of coupled lines. They analyzed and modeled the crosstalk effects manifested as a pulse on the line whose input is held constant. Crosstalk speedup and slowdown effects, not addressed in these papers, were considered by Breuer [36]. In [35], they modeled all CMOS logic gates as equivalent inverters and then calculated the output response of crosstalk noise through this gate using the transfer function of the equivalent inverter. As shown in Figure 9, by using their simple inverter capacitor model they were able to model all four transient effects of the crosstalk faults.

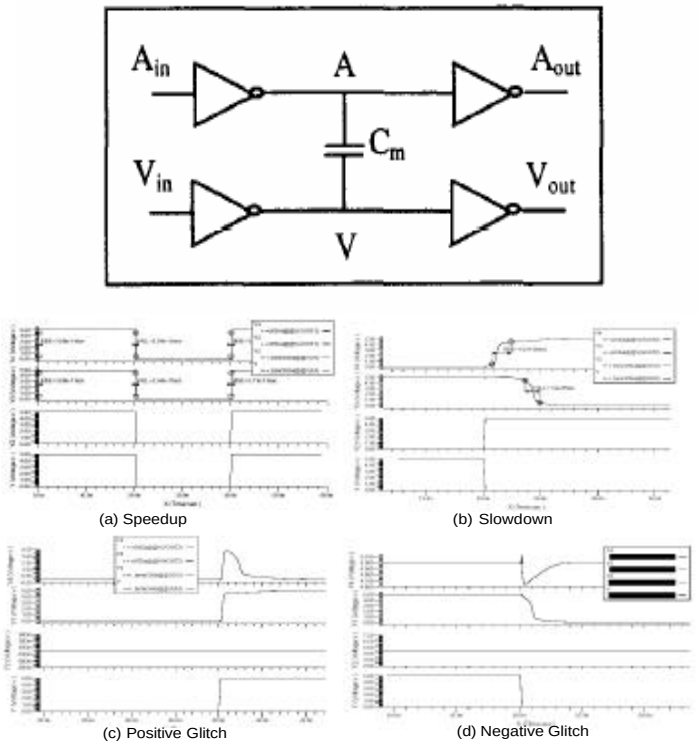

Fig. 9. Inverter-Capacitor model for Crosstalk and 4 transient effects modeled by it

In [35][36] they proposed a mixed signal ATPG algorithm, which used their inverter-capacitor model to generate tests for all four effects of crosstalk fault. The only drawback of this simple mixed signal model for the crosstalk fault is that it is very computational intensive and takes a long time to simulate a small circuit. So, efforts have been made to model crosstalk effects at gate level. Figure 10 shows the first ever-logical level fault model [45] which models crosstalk coupling perturbation effects. It uses a N-stage shift register running at a higher speed clock to detect transition occurring on aggressor (x) line

and puts a glitch pulse on the victim line (y) using simple and-or logic.

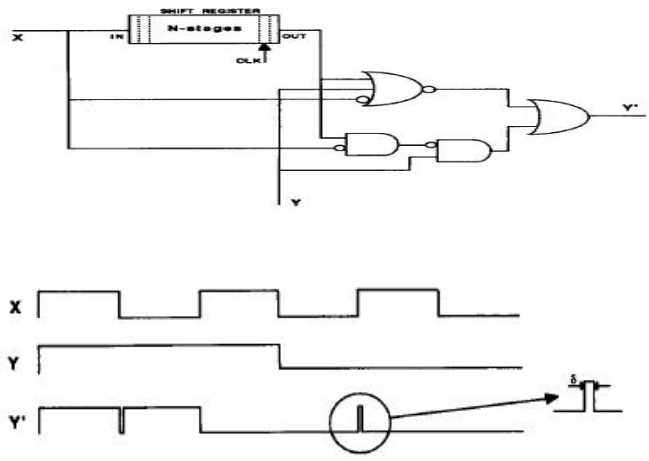

Fig. 10. Logical level model for crosstalk pulse fault using shift registers as transition detector

The width of the crosstalk pulse is controlled by number of stages N of the shift register (which acts as delay elements) and the period T_{clock} of its clock ($\delta = N*T_{clock}$).

Another logic level crosstalk pulse fault is shown in Figure 11. Here XOR gate is used to detect the transition in the aggressor line, i.e., 1->0 or 0->1. Using the logic we can generate positive or negative pulse on victim line. The width of the cross talk pulse could be controlled by changing the number of delays.

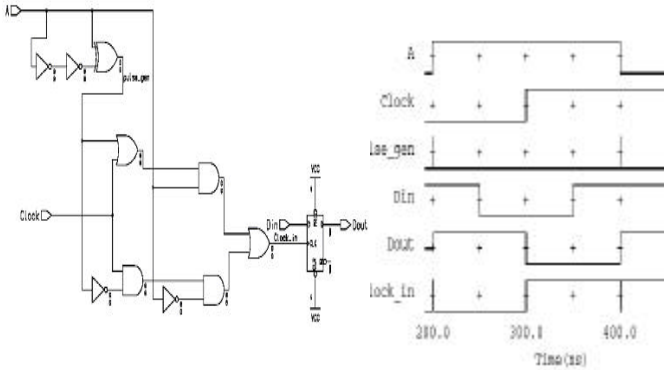

Fig. 11. Logical level model for crosstalk pulse fault using ExOr and inverters as transition detector

Becker [47] proposed a HDL level fault model for crosstalk which is shown in Figure 12 which can be used to generate tests for crosstalk faults in HDL simulators like Cadence's Verifault. Here the wire w1 is assumed to be an aggressor and w2 is assumed to be a victim in a crosstalk defect and the statement **xrf glitch XRF(w2_out,w1, w2)** is included then following behavior will take place: If there is an event (a signal change) on the wire w1, then the internal signal active will be set to 1 for the duration of gltime nanoseconds. Note that the parameter gltime has a default value of 2-unit time, but it can be set for each instance of xrf glitch to different values from outside of the module. For instance, if there is an instance XRF1 and an instance XRF2, one can set XRF1.gltime to 5 and XRF2.gltime to 15. After gltime nanoseconds, active is set back to 0. The victim signal w2 is forced to assume the value of w1 for the time when value of active is 1. The last statement in module's code controls this.

```
module xrf_glitch(vict_out,aggr,vict);
input aggr,vict;
output vict_out;
parameter gltime=2;
reg active;
always @(aggr)
begin
   active=1;
   #(gltime) active=0;
end
assign vict_out=(active)? aggr:vict;
endmodule
```

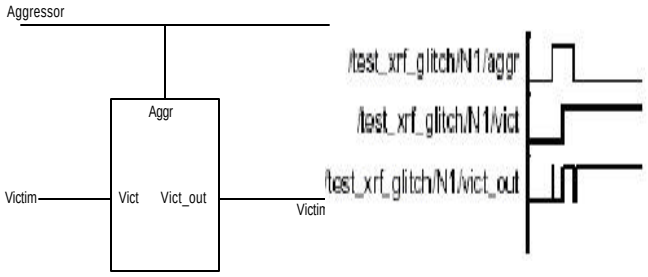

Fig. 12. Verilog code for crosstalk fault model

V. BRIDGING FAULT

Bridging fault can be defined as a short between two nodes, which are otherwise unconnected. These short can be a strong short with a very low resistance or a weak short with a high resistance. Thus the bridging faults have quite different electrical and logic response, which makes their simulation and detection difficult. They can be considered as extension of stuck-at models. Like stuck-at fault, these faults are generated due to problems in fabrication process such as mask misalignment, over-etching, under-etching, etc. It has been found that 30–40 % of faults occurring in IC are Bridging faults [48] and so they have been most looked for faults after stuck-at.

There are three basic methods which are widely used to detect bridging faults, viz. current testing, logic testing, propagation delay testing. Among all these current testing has been considered to be most efficient in detecting bridging faults.[49]

In normal operation of a CMOS circuit only gates are connected to VDD and GND, and the high resistance of these gates prevents dc current flow .The conducting path between VDD and GND exits for only short time during the switching operations. But in the presence of the bridging fault, a conducting path between VDD and GND might exist for all time, causing large amount of current to flow [50]. So presence

of excessive current consumption shows the presence of the bridging fault. Current testing method has been considered as one of the most robust and effective way to detect the bridging fault. But in submicron technology value of leakage current has increased making it difficult to set the threshold for fault detection, and so current testing method is losing its popularity.

The presence of one or more bridging fault causes an extra delay to be introduced into the circuit, so various propagation delay measurement methods could be used to detect bridging faults. Gate propagation delay testing, Path propagation delay testing and direct electrical simulations are three commonly used propagation delay-testing methods to detect the bridging faults. There is an accuracy verses speed trade off while choosing one of these methods. Direct electrical simulation is the most accurate but it takes a large amount of time to simulate. While gate and path delay methods are faster, their accuracy is limited. The gate delay method [51][52] can be used to directly detect particular bridging fault, but path delay method detects classes of bridging faults. Also, path delay method may detect multiple less severe bridging faults, which might go unnoticed in the gate delay technique.

Wired-Or and Wired-And are the most popular logical fault models used to describe bridging fault behavior. Figure 13 shows the behavior of these models. Here a resistive short between two wires is modeled as either a logical AND for wired-AND fault or a logical OR for a wired-OR fault. A and B represents the signal sources for the two nets while A' and B' represents the respective destinations for two nets. As can be seen from table 3, in case of a wired-AND fault if one of the sources is '0' then both the destinations would have value '0'. Similarly, in a wired-OR fault if one of the sources were '1' then both the destinations would have a value '1'. A test vector t could detect wired-AND fault if and only if either it detects A stuck-at-0 and sets B =0, or t detects B stuck-at-0 and sets A=0. Similarly, A test vector t could detect wired-OR fault if and only if either it detects A stuck-at-1 and sets B =1, or t detects B stuck-at-1 and sets A=1 [54]. Thus it is possible to detect simple bridging fault model with any single stuck-at fault test set. [53]. Figure 14 shows simple verilog code, which could model Wire-or and Wired behavior of bridging fault at HDL level. This simple looking Wired-or and Wired-And model which were initially developed for bipolar logic [53] is no longer valid in present day deep sub micron CMOS technology. The nature of the bridge depends upon the resistance of the bridge, pull up/down strength of the gate, threshold voltage of input of gate and location of the bridge [55].

A dominant bridging fault model as shown in Figure 15 was introduced. Here it is assumed that one of the sources has a stronger drive and it determines logic value at both destinations. Becker [47] gave a HDL level model for this dominant bridging fault which is shown in Figure 16. From tables 3 and 4 you could see that dominant bridging fault is more difficult to observe and detect than wired-and and wired-or models. The Wired-And and Wired-or fault can be detected by any one of four conditions:

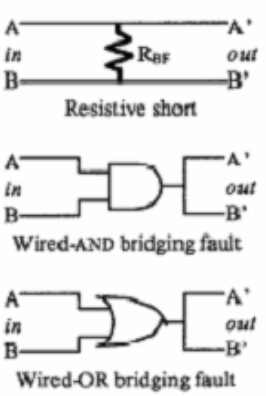

Fig. 13. Wired-And and Wired-OR Fault Models

TABLE III
LOGICAL BEHAVIOR OF WIRED-AND/OR MODELS

In	Out	Wired-AND	Wired-OR
AB	A'B'	A'B'	A'B'
00	00	00	00
01	01	00	11
10	10	00	11
11	11	11	11

```
module wandor (a_bar,b_bar,a,b,sel);
input a,b,sel;
output a_bar,b_bar;

and G11 (andab,a,b);
or G1 (orab,a,b);

assign a_bar = (sel)?orab : andab;
assign b_bar = (sel)?orab : andab;
endmodule
```

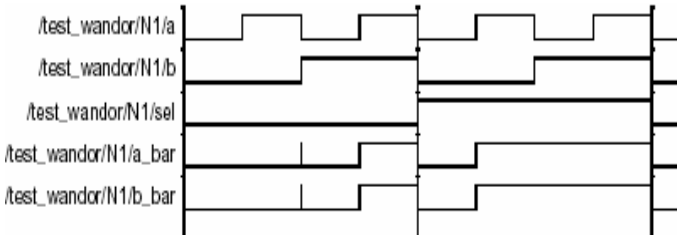

Fig. 14. Verilog Code for Wired-AND and Wired-OR fault model.

1. Apply test vector A=0 and B=1 while observing both A' and B'
2. Apply test vector A=1 and B=0 while observing both A' and B'
3. Apply both test vectors, 01 & 10, while observing only A'
4. Apply both test vectors, 01 & 10, while observing only B'

But for detection dominant bridging fault we need either condition 1 or condition 2. As the detection criteria for the dominant bridging fault is a subset of the detection criteria for the wired-and/ wired-or bridging fault models, a set of test vectors which detect both dominant bridging faults associated

with a given pair of nets could also detect both wired-and and wired-or bridging faults.

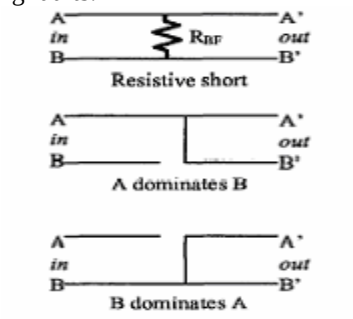

Fig. 15 . Dominant Bridging Fault Models

TABLE IV
LOGICAL BEHAVIOR OF DOMINANT MODELS

In	Out	A dom B	B dom A
AB	A'B'	A'B'	A'B'
00	00	00	00
01	01	00	11
10	10	11	00
11	11	11	11

Although good fault coverage for bridging fault can be obtained using the dominant bridging fault model, yet behavior of some actual manufacturing defects could not be accurately

```
module dominant
(aggr,victim,aggr_out,victim_out);
input aggr,victim;
output aggr_out,victim_out;

assign aggr_out = aggr;
assign victim_out = aggr;

endmodule
```

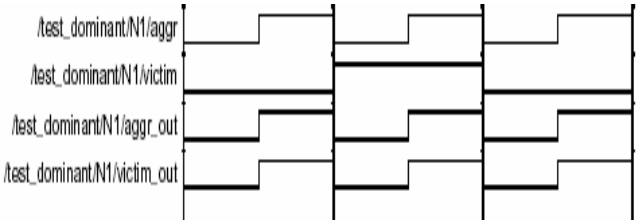

Fig. 16. Verilog Code for Dominant Bridging fault model

model by either of the two bridging fault model and they escape fault detection with tests developed for bridging faults. To take into account of these effects of bridging faults a new model diode-And or dominant-And and diode-or or dominant-Or has been introduced [54]. As shown in Figure 17 this model can be seen as a hybrid of the wired-And/wired-or and dominant bridging fault models. From table 5 it is noticed that the detection of all four possible fault models between a given pairs of nets requires both test vectors (01 and 10) as well as observation of both destination nets (A' and B'). As a result, this bridging fault model is more difficult to detect than either wire-and/wire-or or the dominant bridging fault models and take more time to simulate, as now there are 4 cases to simulate. However, detection of the four-dand/dor fault model ensures that all wired-And/wired-Or and dominant bridging fault models will also be detected for given pair of nets.

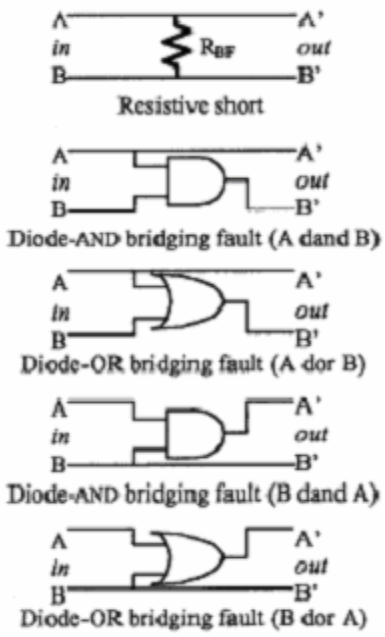

Fig. 17. Diode-And and Diode-Or Bridging Fault Models

TABLE V
LOGICAL BEHAVIOR OF DIODE-AND AND DIODE-OR BRIDGING FAULT MODELS

In	Out	A dand B	A dor B	B dand A	B dor A
AB	A'B'	A'B'	A'B'	A'B'	A'B'
00	00	00	00	00	00
01	01	00	01	01	11
10	10	10	11	00	10
11	11	11	11	11	11

VI. CONCLUSION

In this paper, research on contemporary fault models was surveyed. There is a wealth of information on this topic, and one survey paper cannot cover all pertinent topics. We hope that this paper will provide a reference material for someone interested in learning more about fault models. For details that were omitted here, the reference list would be of use.

VII. ACKNOWLEDGMENT

The authors would like to thank Professor Kewal Saluja of the Department of Electrical Engineering, UW – Madison, for valuable suggestions during the course of the project.

REFERENCES

[1] W. Y. Koe and S. F. Midkiff, "Circuit simulation of CMOS faults," *Southeastcon '88, IEEE Conference Proceedings*, pp. 87-91, 1988.

[2] J. A. Abraham and W. K. Fuchs, "Fault and error models for VLSI," *Proc. of the IEEE*, pp. 639-654, 1986.

[3] B. Ricco, M. Favalli, and P. Olivo, "Comprehensive fault modeling and simulation in CMOS ICs," *CompEuro '91, 5th Annual European Computer Conf. Proc.*, pp. 778-785, May 1991.

[4] R. L. Wadsack, "Fault modeling and logic simulation of CMOS and NMOS integrated circuits," *Bell Syst. Tech J.*, vol. 57, pp. 1449-1474, 1978.

[5] S. Al-Arian and D. Agrawal, "Modeling and testing of CMOS circuits," *Proc. of IEEE Int. Conf. On Computer Design*, pp. 763-769, 1984.

[6] S. Reddy, V. Agrawal, and S. Jain, "A gate level model for CMOS combinational logic circuits with application to fault detection," *Proc. of Design Aut. Conf.*, pp. 504-509, 1984.

[7] C. Lo, H. Nham, and A. Bose, "Algorithms for an advanced fault simulation system in MOTIS," *IEEE Trans. on CAD*, vol. 6, pp. 232-240, 1987.

[8] C. C. Beh, K. H. Arya, C. E. Radke, and K. E. Torku, "Do stuck fault models reflect manufacturing defects?," *Proc. IEEE Semiconductor Test Conf.*, pp. 35-42, Nov. 1982.

[9] J. Alt and U. Mahlstedt, "Simulation of non-classical faults on the gate level – Fault modeling," *VLSI Test Symp.*, pp. 351-354, 1993.

[10] M. Favalli, P. Olivo, M. Damiani, B. Ricco, "Fault simulation of unconventional faults in CMOS circuits," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, pp. 677-682, 1991.

[11] A. K. Majhi and V. D. Agrawal, "Delay fault models and coverage," *Proc. of VLSI Design*, pp. 364-369, 1998.

[12] K. T. Cheng, "Transition fault simulation for sequential circuits," *Proc. International Test Conference*, pp. 723-731, Oct. 1992.

[13] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition fault simulation", *IEEE Design and Test of Computers*, 4:32-38, 1987.

[14] R. C. Aitken, "Nanometer technology effects on fault models for IC testing," *IEEE Computer Magazine*, pp. 46-51, 1999.

[15] J. L. Carter, V. S. Iyengar, and B. K. Rosen, "Efficient test coverage determination for delay faults," *Proc. International Test Conf.*, pp. 418-427, 1987

[16] A. K. Pramanick and S. M. Reddy, "On the detection of delay faults," *Proc. 1988 IEEE Int. Test Conf.*, pp. 845-856, 1988

[17] V. S. Iyengar, B. K. Rosen, and I. Spillinger, "Delay test generation 1 – Concepts and coverage metrics," *Proc. 1988 IEEE Int. Test Conf.*, pp. 492-499, 1988

[18] V. S. Iyengar, B. K. Rosen, and J. A. Waicukauski, " On computing the sizes of detected delay faults," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 299-312, 1990

[19] G. L. Smith, "Model for delay faults based upon paths," *Proc. Int. Test Conf.*, pp. 342-349, 1985

[20] W. K. Lam, A. Saldanha, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Delay fault coverage, test set size, and performance trade-offs," *IEEE Trans. on CAD*, 14(1): 32-44, 1995

[21] K. T. Cheng and H. Chen, "Delay testing for non-robust untestable circuits," *Proc. Int. Test Conf.*, pp. 723-731, 1992

[22] K. T. Cheng and H. C. Chen, "Classification and identification of non-robust untestable path delay faults," *IEEE Trans. on CAD*, 15: 845-853, 1996

[23] M. A. Gharaybeh, M. L. Bushnell, and V. D. Agrawal, "Classification and test generation for path-delay faults using single stuck-at fault tests," *Jour. Electronic Testing: Theory and Applications*, 11(1): 55-67, 1997

[24] M. Sivaraman and A. J. Strojwas, "Primitive path delay fault identification," *Proc. 10th Int. Conf. on VLSI Design*, pp. 95-100, 1997

[25] K. Heragu, J. H. Patel, and V. D. Agrawal, "Segment delay faults: A new fault model," *Proc. 14th IEEE VLSI Test Symp.*, pp. 32-39, 1996

[26] T. J. Chakraborty, V. D. Agrawal, and M. L. Bushnell, "Delay fault models and test generation for random logic sequential circuits," *29th ACM/IEEE DAC*, pp. 165-172, 1992

[27] P. Uppaluri, I. Pomeranz, and S. M. Reddy, "Test pattern generation for path delay faults in synchronous sequential circuits using multiple fast clocks and multiple observation times," *24th Int. Symp. on Fault Tolerant Computing*, pp. 456-465, 1994

[28] X. Xie and A. Albicki, "New advances in path delay fault testing of combinational circuits," *VLSI Test Symp.*, pp. 272-277, 1994.

[29] T. J. Chakraborty and V. D. Agrawal, "Effective path selection for delay fault testing of sequential circuits," *Proc. Int. Test Conf.*, pp. 998-1003, 1998

[30] D. H. Du, S. H. Yen, and S. Ghanta, "On the general false path problem in timing analysis," *Proc. ACM/IEEE Design Automation Conf.*, pp. 555-560, 1989

[31] W. B. Jone, W. S. Yeh, C. Yeh, and S. R. Das, "An adaptive path selection method for delay testing," *IEEE Trans. on Instrumentation and Measurement*, vol. 50, pp. 1109-1118, 2001.

[32] S. Ohtake, K. Ohtani, H. Fujiwari, "A method of test generation for path delay faults using stuck-at fault test generation algorithms," *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, pp. 310-315, 2003.

[33] A. Saldanha, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "Equivalence of robust delay fault and single stuck-fault test generation," *Proc. of Int. Conf. on Computer-Aided Design*, pp. 418-421, 1992.

[34] Weiyu Chen; Gupta, S.K.; Breuer, M.A.; "Analytic models for crosstalk delay and pulse analysis under non-ideal inputs" in Test Conference, 1997. Proceedings., International , 1-6 Nov 1997 Page(s): 809 -818

[35] Weiyu Chen; Gupta, S.K.; Breuer, M.A.; "Test generation in VLSI circuits for crosstalk noise" in Test Conference, 1998. Proceedings. International, 18-23 Oct 1998 , Page(s): 641 –650

[36] Wei-Yu Chen; Gupta, S.K.; Breuer, M.A.; "Test generation for crosstalk-induced delay in integrated circuits", in Test Conference, 1999. Proceedings. International , 1999, Page(s): 191 –200

[37] Shimizu, K.; Takamura, M.; Shirai, T.; Itazaki, N.; Kinoshita, K.; "Fault simulation method for crosstalk faults in clock-delayed domino CMOS circuits", in The First IEEE International Workshop on Electronic Design, Test and Applications, 2002. Page(s): 92 –96

[38] Itazaki, N.; Idomoto, Y.; Kinoshita, K.; "A fault simulation method for crosstalk faults in synchronous sequential circuits", in Fault Tolerant Computing, 1996., Proceedings of Annual Symposium on , 25-27 Jun 1996, Page(s): 38 –43

[39] Takahashi, H.; Phadoongsidhi, M.; Higami, Y.; Saluja, K.K.; Takamatsu, Y.; "Simulation-based diagnosis for crosstalk faults in sequential circuits" in Test Symposium, 2001. Proceedings. 10th Asian , 2001, Page(s): 63 –68

[40] Keller, K.J.; Takahashi, H.; Saluja, K.K.; Takamatsu, Y.; "On reducing the target fault list of crosstalk-induced delay faults in synchronous sequential circuits", in Test Conference, 2001. Proceedings. International , 2001, Page(s): 568 –577.

[41] Keller, K.J.; Takahashi, H.; Le, K.T.; Saluja, K.K.; Takamatsu, Y.; "Reduction of target fault list for crosstalk-induced delay faults by using layout constraints" in Test Symposium, 2002. (ATS '02). Proceedings of the 11th Asian , 2002, Page(s): 242 –247.

[42] Zain, A.E. ; Chowdhury, S.; "An analytical method for finding the maximum crosstalk in lossless-coupled transmission lines." In Int'l Conf. On Computed Aided Design, pp.443-448, 1992

[43] Gordon, C.; Rossele, K.M.; "Estimating Crosstalk in multiconductor transmission lines", in IEEE Trans. On Components Packaging and Manufacturing Technology, Vol 19, pp. , May1996.

[44] Kaupp, R.; "Waveform degradation in VLSI interconnections" in IEEE Journal of Solid-State Circuits,Vol24, pp. 1150-1153, August 1989.

[45] Rubio, A.; Itazaki, N.; Xu, X.; Kinoshita, K.; "An approach to the analysis and test of crosstalk faults in digital VLSI circuits", in Design Automation. EDAC. Proceedings of the European Conference on , 25-28 Feb 1991, Page(s): 72 –79

[46] Moll, F.; Rubio, A.; "Methodology of detection of spurious signals in VLSI circuits", in European Test Conference, 1993. Proceedings of ETC 93., Third , 19-22 Apr 1993 Page(s): 491 –496.

[47] Bradford, J.; Delong, H.; Polian, I.; Becker, B.; "Simulating realistic bridging and crosstalk faults in an industrial setting " in European Test Workshop, 2002. Proceedings. The Seventh IEEE , 2002, Page(s): 75 –80

[48] Shen, J.P.; Maly, W.; Ferguson, F.J; "Inductive fault analysis of MOS integrated circuits" in IEEE Design & Test, pp 13-26, 1985

[49] Ryan, C.A.; "On the complexity of bridging fault simulation techniques for CMOS integrated circuits" in ASIC Conference and Exhibit, 1995., Proc of 8th Annual IEEE International , 18-22 Sep 1995 Page(s): 160 –163

[50] Nigh, P.; Maly, W.; "Test generation for current testing ", in European Test Conference, 1989., Proceedings of the 1st , 12-14 Apr 1989 Page(s): 194 –200

[51] Carter, J; Iyenger, V; Rosen, B; "Efficient Test Coverage Determination for Delay Faults" in Proc. ITC, pp 418-427,1987

[52] Mao, W; Ciletti, M; "A simplified Six-Waveform type method for delay fault testing" in 26th ACM/IEEE DAC, pp 730-733 1989.

[53] K.C.Y. Mei, "Bridging and Stuck-At faults" IEEE Trans. On computer, Vol 23, No. 7, pp720-727, 1974

[54] Emmert, J.M.; Stroud, C.E.; Bailey, J.R.; "A new bridging fault model for more accurate fault behavior" in AUTOTESTCON Proceedings, 2000 IEEE , 2000 Page(s): 481 –485

[55] Favalli, M.; Dalpasso, M.; "Bridging fault modeling and simulation for deep submicron CMOS Ics", in Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , Volume: 21 Issue: 8 , Aug 2002 Page(s): 941 –953