A Gate-Level Method for Transistor-Level Bridging Fault Diagnosis

Xinyue Fan, Will Moore
Department of Engineering Science
Oxford University, OX1 3PJ
{xinyue.fan, will.moore}@eng.ox.ac.uk

Camelia Hora, Mario Konijnenburg, Guido Gronthoud
Philips Research Labs
Prof. Holstlaan 4, 5656AA Eindhoven, Netherlands
{camelia.hora, mario.konijnenburg, guido.gronthoud}@philips.com

Abstract

The paper addresses the issue of transistor-level bridging fault diagnosis. While most of the previous bridging fault diagnosis work focuses on the gate-level bridging faults, this method provides a solution to intra-gate bridging faults diagnosis for the first time. Instead of using any transistor level simulation tools, we develop a transformation technique that allows transistor-level bridging faults to be diagnosed by the commonly used gate-level bridging faults diagnosis tools. Real diagnosis results from Philips designs are presented.

1. Introduction

As the new IC manufacturing technology continues to offer higher integration density, manual search and identification of the defects becomes impractical because of the exceedingly high cost and the time it could take. Therefore automatic fault diagnosis is needed in order to provide a short-list of candidate defects for precise failure analysis.

Bridging faults are among the most commonly seen defect types in IC manufacturing [1]. Inter-gate bridging faults have been extensively studied and methods for their diagnosis have been developed [2] [3] [4] [7] [8]. The basic approach used is to build up composite bridging fault signatures from stuck-at fault signatures by assuming a certain bridging fault model. The effect of the electrical shorts has been assumed equal to an AND gate or an OR gate, known as the wired-AND and wired-OR bridging fault model. Another straightforward bridging fault model is the dominant model, in which one of the shorted lines always drives the value of the other. More accurate models like the voting model and the bias voting model are proposed in [12] [13]. These bridging fault models focus on the logical behaviour of the fault and thus are easy to implement in simulation, test pattern generation and diagnosis tools. The other main approach for bridging fault diagnosis is the application of Iddq testing, which makes use of the increased quiescent current generated when the component nodes of the bridge are driven to opposite values [5] [6].

All these methods are implemented at the inter-gate level. However, since complex gates are often used in modern CMOS design, the chances of a bridging fault occuring as an intra-gate fault are high. The gate-level diagnosis tools are unable to handle this type of fault as the bridged intra-gate nodes are not represented in the netlist. Up until now, no systematic diagnosis method has been given to tackle intra-gate bridging faults. In this paper, we introduce a method that can migrate the commonly used inter-gate level bridging fault model to the intra-gate level, thus enabling a commercial bridging fault diagnosis tool to diagnose the bridging faults inside gates.

The paper is organized as follows. Section 2 discusses the way that suspected intra-gate bridging gates are short-listed. In section 3, a new transformation method is proposed and the rules to represent the intra-gate bridging faults in the inter-gate level are discussed. Section 4 is the overall flow of our diagnosis method. Experimental results and analysis are given in Section 5.

2. Shortlist the possible gates with intra-gate bridging faults

Before we introduce the new method, some terminologies and the basic diagnosis flow should be made clear. The diagnosis flow is literally a process of comparing what we see on the tester and what we have predicted by simulation of the assumed fault.

Here we define two sets – *Obs* and *Sim*. The set *Obs* (Observed Results) represents the pairs of pattern/failed pin observed on the tester, and the set of *Sim* (Simulation Results) represents the pairs predicted by the fault simulation.

There are two different versions of the diagnostic measures used to determine the quality of the diagnosis [3] [9], but both involve ranking candidate faults by their values of *Obs* and *Sim*. In this paper, we use the system defined by Hora [3] who uses two measures to judge the quality of the diagnosis. The term "Matching" quantifies

Proceedings of the 24th IEEE VLSI Test Symposium (VTS'06)
0-7695-2514-8/06 $20.00 © 2006 IEEE

the extent to which the observed failing results actually match with the simulation of a particular fault.

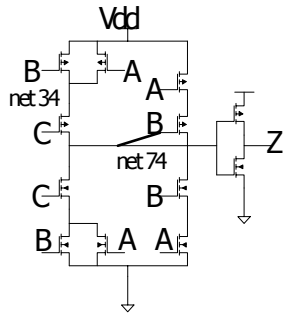
$$\text{Matching (M)} = \frac{|Obs \cap Sim|}{|Obs|} \times 100\%$$

A second term "Prediction" quantifies the extent to which simulations of a particular fault predict *only* those failing observed results.

$$\text{Prediction (P)} = \frac{|Obs \cap Sim|}{|Sim|} \times 100\%$$

Because we cannot realistically make an intra-gate bridging diagnosis for every gate in a large circuit, a shortlist of possible faulty gates, in which an intra-gate bridging fault may be present, is generated by using a net diagnosis model.

The net diagnosis model, initially developed for interconnect open defects [10], combines both stuck-at-0 and stuck-at-1 signatures of a net, thus guaranteeing that the output of the faulty gate will have Matching=100%. Consider the circuit in Figure 1 and the patterns shown in Table 1. We assume a net diagnosis model on the output line Z.



Pattern	ABC	Z
#1	000	0
#2	001	0
#3	010	0
#4	011	1
#5	100	0
#6	101	1
#7	110	1
#8	111	1

Figure 1. *Sample gate* ***Table 1.*** *Sample patterns*

The Z stuck-at-1 signature is {#1 (Z fails at pattern #1) , #2, #3, #5} and the Z stuck-at-0 signature is {#4, #6, #7, #8}, so the net signatures of Z is *Sim* = {#1, #2, #3, #4, #5, #6, #7, #8}. Suppose the resistance of the bridging fault is 1 ohm between B and net74, electrical simulation is performed to generate the observed result, *Obs* = {#1, #2, #4, #5, #7}. According to the definition of Matching and Prediction:

$$\text{Matching(M)} = \frac{|Obs \cap Sim|}{|Obs|} \times 100\% = \frac{|\{\#1,\#2,\#4,\#5,\#7\} \cap \{\#1,\#2,\#3,\#4,\#5,\#6,\#7,\#8\}|}{|\{\#1,\#2,\#4,\#5,\#7\}|} \times 100\% = 100\%$$

$$\text{Prediction(P)} = \frac{|Obs \cap Sim|}{|Sim|} \times 100\% = \frac{|\{\#1,\#2,\#4,\#5,\#7\} \cap \{\#1,\#2,\#3,\#4,\#5,\#6,\#7,\#8\}|}{|\{\#1,\#2,\#3,\#4,\#5,\#6,\#7,\#8\}|} \times 100\% = 63\%$$

Clearly the output net will always have Matching=100% under the net diagnosis model even though the fault is actually inside the gate. However we expect a low Prediction since no real defect will cause the output net both stuck-at 0 and stuck-at 1.

The diagnosis tools working under net diagnosis model will pick out those nets with Matching = 100% but Prediction < 100%, and our method will further investigate these gates whose output nets have been picked out. If there is real intra-gate bridging fault which matches with one of the models we used, the result of our method will be both Matching=100% and Prediction=100%, the most convincing diagnosis result.

Some of the intra-gate bridging faults will only see fault signatures either from output stuck-at-1 or from output stuck-at-0 appearing in its *Obs*. For instance in Figure 1, when net34 is bridged with Vdd (the fault can be also modeled as net34 stuck-at-1), we would only see sometimes net74 stuck-at-1 and, consequently, Z stuck-at-0. So, for those faults, the faulty gates will be diagnosed under stuck-at model as Matching = 100%, Prediction < 100%.

Therefore, to cover all the intra-gate bridging faults, those gates with output diagnosed as Matching = 100%, Prediction < 100% under either net diagnosis model or stuck-at model should be taken as our primary intra-gate bridging fault candidates.

3. Transform the transistor schematics

Having short-listed the primary candidate gates, we need to consider how to confirm if the gates contain intra-gate bridging faults as well as which bridging faults. The most direct solution would be to use electrical level simulation to predict the fault signatures (*Sim*) of every possible intra-gate bridging pairs and then compare them with the *Obs* we extract on the gate output. However, the number of all the possible intra-gate bridging pairs and the numerous patterns we have to simulate means time may become an issue. Also, for every simulation, we have to assume a certain resistance value which does not guarantee that the simulation result will be the same as for the real defect. In addition, the whole diagnosis software has to be rewritten.

We believe the traditional bridging fault diagnosis tools should not be restricted in the gate-level and we can extend its capability to the intra-gate domain. To use the gate-level bridging fault model for a transistor-level circuit, we have to transform the transistor-level

description to the gate-level while having all the potential bridging nets represented. (Some of the fault simulation tools have the transistor-level description, however the transistor primaries are normally unidirectional, thus they are not able to represent the values of the internal nodes.) The transformed gate can then be directly diagnosed with the traditional bridging fault diagnosis tools.

The gate-level bridging diagnosis tools analyze the values of possible bridging nets under each pattern. If the values between two possible bridging nets are different, then certain stuck-at fault signatures, depending on the bridging model used, are taken as the bridging fault signatures [2]. To fit in with this process, during the course of transformation, we follow two principles:

1) Every net (for instance in Figure 2: A, B, C, D, E, net1, net2, net3, net4, net5, Z) in the transistor-level schematics will be represented in the transformed circuits, and their values at every test pattern will remain the same as those before the transformation.
2) If a stuck-at effect propagates to the output i.e. it is not blocked by any off transistors, the corresponding stuck-at fault in the transformed circuit will also propagate.

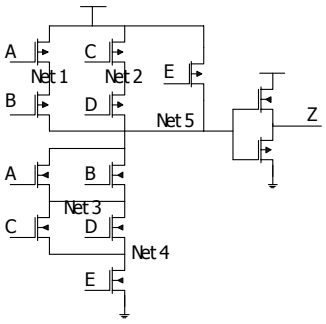


Figure 2. *The gate to be transformed*

Now we demonstrate the transformation method through the example gate in Figure 2. There are five internal nets in this particular gate (Net1-5). According to our first principle, all the five internal nets must be represented in a gate-level description where their values are kept. The following rules are set to justify the values of the internal nets.

1. For n-transistors part (Net3 and Net4), because the transistors are bi-directional, we need to do two rounds of justification, one from the GND and one from the net connecting the n-transistors and p-transistors, in this case Net5. In each round of justification, we have the following steps.

(a) Replace all the n-transistors with the element as shown in Figure 3. The purpose is to guarantee that the zero value from the source will be transmitted to the drain when value on the gate is one (transistor turned-on).

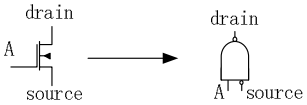


Figure 3. *Replacement of n-transistor*

(b) Replace with an AND gate where a parallel connection between transistors is present, as Figure 4. The AND gate makes sure if one of the drains is zero, the output will be zero, thus guaranteeing the propagation of zero value from the source.

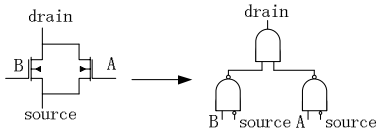


Figure 4. *Replacement of parallel n-transistors*

Figure 5 shows the view after two rounds of justifications. We also need to generate Net5_c, the correct value of net5, by logic combinations, as it will be later used to justify the internal nets. The rules to generate Net5_c are same as the rules we used in the first round of justification (the one from the GND). We use a similar approach to generate the correct values for the junction nets of p-transistors and n-transistors in [11], where a detailed explanation can be found.

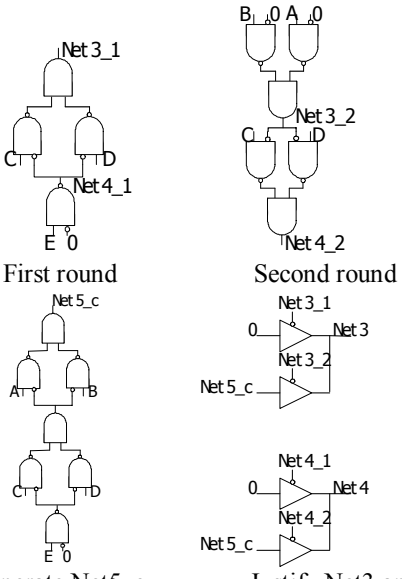


Figure 5. *The transformation of N transistors*

Net3_1 and Net4_1 in the first round justification respectively indicate whether Net3 and Net4 are electrically connected to the GND. A zero value shows it is connected, a one value shows it is disconnected.

Likewise, to judge if Net3 and Net4 are electrically connected to Net5, we have to look at the value of Net3_2 and Net4_2. These four nets with underscore function as the connection indicators.

The final justification for Net3 and Net4 is done by the four tri-state bus drivers, each controlled by a connection indicator and each individually decides if the value of GND or Net5 should be loaded to the Net3 and Net4. Up to this stage, Net3 and Net4 are fully justified.

2. For the p-transistor part (Net1 and Net2), we also have to perform two rounds of justification, one from the Vdd and one from Net5. The rules of transformation are similar to those for the n-transistor part, except when replacing the p-transistor, we use the element in Figure 6 instead of the one in Figure 3.

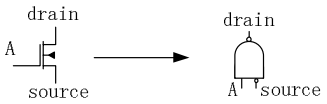


Figure 6. *Replacement of p-transistor*

Figure 7 shows the transformation of P transistors to justify the internal nets, Net1 and Net2. Note that there is no need to transform the transistor that directly connects either Vdd or GND to Net5, like the p-transistor control by E, for there is no internal net.

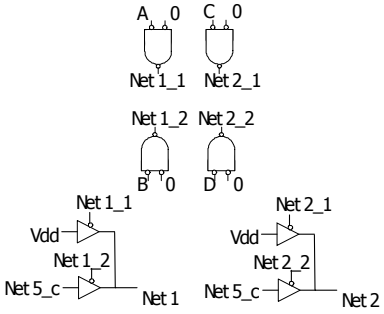


Figure 7. *The transformation of P transistors*

Now the values of internal nets have been fully justified. Next, as required by our second principle, we have to connect these internal nets in a way that all the stuck-at signatures are propagated when they are not completely blocked by the off transistors. Because we already have the connection indicators of every internal net (Net1_2, Net2_2, Net3_2, Net4_2) to Net5, the only thing we need to do is to add tri-state bus drivers controlled by the connection indicators, whose values decide if a stuck-at signature would be propagated to Net5 and subsequently to the output Z. Figure 8 is the whole view after the final transformation. Transistors like the p-transistor controlled by E in Figure 2, which directly connect Vdd or GND to the p-n junction net, have to be replaced by tri-state bus drivers to help Net1-4 to justify the value of Net5. Note that we have added a tri-state bus driver controlled by E.

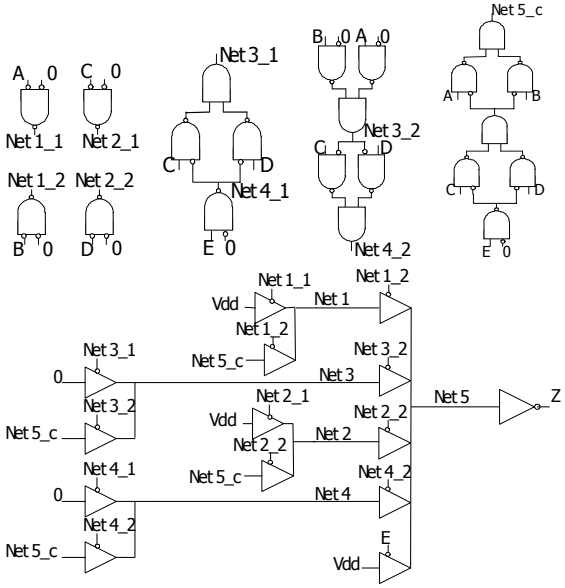


Figure 8. *Final transformation*

Now the two principles are met after the final transformation, the values of Net1-5 are kept and the stuck-at signatures of each internal net are duly propagated when condition allows.

4. Overall flow of diagnosis

The diagnosis flow can be summarized as four steps illustrated in Figure 9. First, a preliminary stuck-at fault diagnosis and failing net diagnosis is performed to shortlist the possible faulty gates with intra-gate bridging. Those gates whose outputs are diagnosed as Matching = 100% and Prediction < 100% under either stuck-at model or net diagnosis model are the primary suspects. The second step is to transform these gates according to the rules set in Section 3. (If desired, transformations can be precomputed for every cell in the library.) The purpose is to use the gate-level bridging diagnosis tools to diagnose possible bridging pairs in the transformed gates. But to achieve that, we do not have to perform the bridging diagnosis on the whole circuit. Instead, in the third step, for those patterns that propagate the failure signatures of the gate's output, we extract the input values of this particular gate and compose them into new patterns for the transformed gate. In the final step, the gate-level bridging diagnosis is applied on this transformed gate only, which takes very little time.

We do not yet have a tool to extract realistic intra-gate bridging pairs from the gate layout, though it would be useful in order to enhance the diagnostic resolution. Currently all pairs of two nets are set as possible intra-

gate bridging faults when we performed the final gate-level bridging diagnosis.

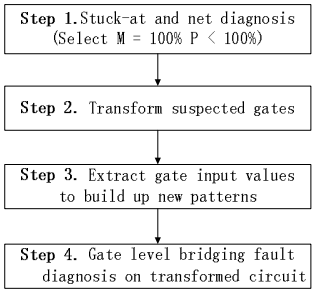


Figure 9. *Overall flow*

5. Experimental results

Experiments are performed with a Philips internal diagnosis tool - FALOC, which has the capacity of stuck-at fault, net fault and bridging fault diagnosis. The wafer testing data are from three different Philips' designs. We have shown seven successfully diagnosed intra-gate bridging faults in Table 2. The second and the third column are the type of the faulty gate and the intra-gate bridging fault it has. The fourth column shows the Matching and Prediction results of this particular gate when the first round diagnosis is performed (Step 1). After the transformation, the second round diagnosis is performed and the results are given in column five. The last column shows the number of intra-gate bridging faults that have been diagnosed as Matching = 100%, Prediction = 100%. Sometimes the number is more than one, therefore we have to refer to the gate layout to judge if one of them is a realistic bridging fault.

For all the seven faulty dies, we are able to enhance the Prediction to 100% after the gate is transformed. Layout information showed that just one of these is a likely site for a bridge (Die #4, Die #5 and Die #7). For those singly diagnosed bridging fault (Die #1, Die #2, Die #3, Die #6), a likely layout explanation was also found. (This step would be unnecessary should a layout extraction tool for intra-gate bridging faults be available before the second round diagnosis.)

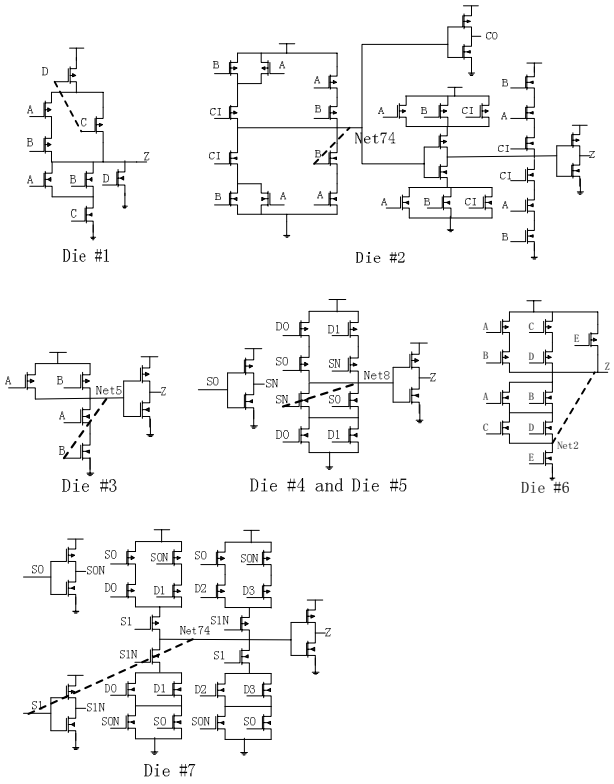
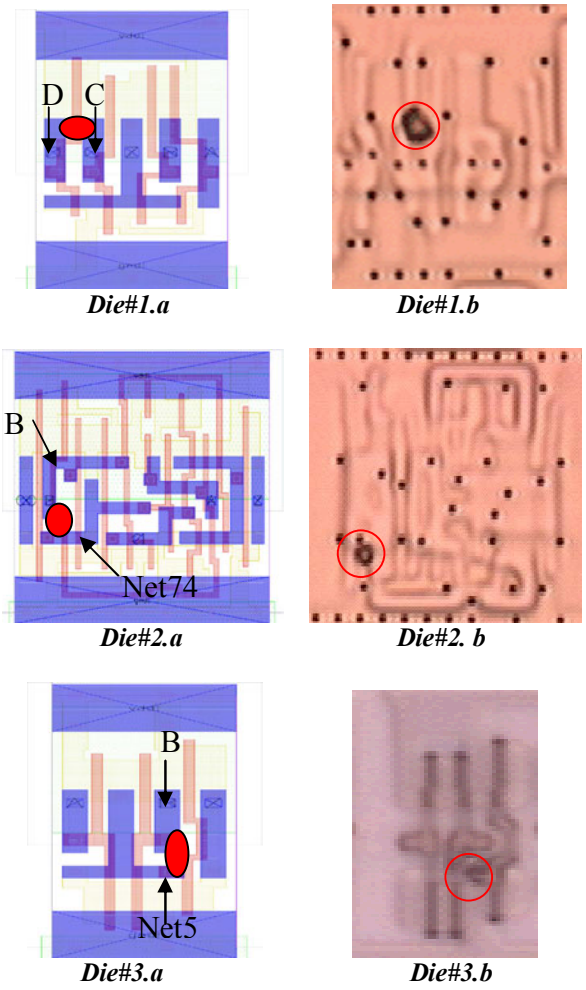


Figure 10. *Seven intra-gate bridging faults*

To further prove our diagnosis, PSPICE-like transistor-level simulations have been performed for all the seven intra-gate bridging faults, and the simulation results confirm the modeled behaviours of all the bridging faults. For Die #1, Die #2 and Die #3, we have been able to get the inline inspection information. Our diagnosis points to bridging faults between C and D (Figure 11, Die1#.a), B and Net74 (Figure 11, Die#2.a), B and Net5 (Figure 11, Die#3.a). The inline inspection pictures in Figure 11.b reveal bridging faults that match exactly with our diagnosis results.

Die	Gate Type	Bridged Nets	M (Matching) and P (Prediction) before transformation	M (Matching) and P (Prediction) after transformation	Number of MP = 100% faults
#1	ao32	C and D	Z stuck-at-1 M = 100% P = 63%	C D wired-AND M = 100% P = 100%	1
#2	fa1	B and Net74	CO net model M = 100% P = 61%	B dominates Net74 M = 100% P = 100%	1
#3	an2	B and Net5	Z net model M = 100% P = 42%	B Net5 wired –AND M = 100% P = 100%	1
#4	mx21	SN and Net8	Z net model M = 100% P = 48%	SN dominates Net8 M = 100% P = 100%	2
#5	mx21	SN and Net8	Z net model M = 100% P = 54%	SN dominates Net8 M = 100% P = 100%	2
#6	ao36	Net2 and Z	Z stuck-at-0 M = 100% P = 3%	Net2 dominates Z M = 100% P = 100%	1
#7	mx41x4	S1 and Net74	Z net model M = 100% P = 33%	S1 dominates Net74 M = 100% P = 100%	2
#7	mx41x4	S1 and Net74	Z net model M = 100% P = 33%	S1 dominates Net74 M = 100% P = 100%	2

Table 2. *Successful diagnosis*



Die#1.a *Die#1.b*

Die#2.a *Die#2. b*

Die#3.a *Die#3.b*

Figure 11. *Failure analysis of Die #1, Die #2, Die #3*

Unfortunately, the other four dies are no longer available for the physical failure analysis. Nonetheless, the diagnosis results have been confirmed by transistor-level simulations.

6. Conclusion

A transformation method is introduced, which allows gate-level bridging diagnosis tools to diagnose intra-gate bridging fault. The method costs little extra time on top of the initial stuck-at and net model diagnosis because the extra steps are performed on the suspected gates only. Gate layout information is used to prune out the unrealistic intra-gate bridging faults. The seven successful diagnosis results, supported by electrical simulation and strengthened by three inline inspection results, prove the effectiveness of this method. This new work, when put together with our previous work on diagnosis of intra-gate stuck-opens [11], provides a powerful extension of our ability to diagnose real defects.

7. Acknowledgement

The authors would like to thank Ananta Majhi, Mohamed Azimane, Maurice Lousberg of Philips Research and Stefan Eichenberger of Philips Semiconductors for their much appreciated help to this work.

[1] F.J. Ferguson and J.P. Shen, "A CMOS fault extractor for Inductive Fault Analysis". *IEEE Transactions on Computer-Aided Design*, Vol 7 (11), 1988, pp. 1181-1194.

[2] D. Lavo et al, "Bridging Fault Diagnosis in the Absence of Physical Information", *International Test Conference,*1997, pp. 887-893.

[3] C. Hora "On Diagnosing Faults in Digital Circuits", PhD Thesis, Technique University Eindhoven, 2002.

[4] S.D Millman et al, "Diagnosing CMOS bridging faults with stuck-at fault dictionaries", *International Test Conference,*1990, pp. 860-870.

[5] S. Chakravarty, s. Suresh, "IDDQ measurement based diagnosis of bridging faults in full scan circuits", *Proceedings of the Seventh International Conference on VLSI Design,*1994 pp.179 – 182.

[6] C. Thibeault, "On the adaptation of Viterbi algorithm for diagnosis of multiple bridging faults", *IEEE Transactions on Computers*, Vol 49, Issue 6, June 2000 pp.575 – 587.

[7] B. Chess et al, "Diagnosis of realistic bridging faults with single stuck-at information", *IEEE/ACM International Conference on Computer-Aided Design*, 1995, pp.185 – 192.

[8] J. Wu, E.M. Rudnick, "Bridge fault diagnosis using stuck-at fault simulation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol 19, Issue 4, April 2000, pp.489 – 495.

[9] S. Venkataraman and S. B. Drummonds, "POIROT: A Logic Fault Diagnosis Tool and Its Applications", *International Test Conference*, 2000. pp. 253-262.

[10] S. Venkataraman and S. B. Drummonds, "A technique for logic fault diagnosis of interconnect open defects", *IEEE VLSI Test Sym*posium, 2000.pp. 313 – 318.

[11] X. Fan et al, "A Novel Stuck-At Based Method for Transistor Stuck-Open Fault Diagnosis", *International Test Conference*, 2005, Accepted.

[12] J. M. Acken and S. D. Millman, "Accurate Modelling and Simulation of Bridging Faults", *Custom Integrated Circuit Conference,* 1991. pp. 63-72

[13] P. C. Maxwell and R. C. Aitken, "Biased Voting: A Method for Simulating CMOS Bridging Faults in the Presence of Variable Gate Logic Thresholds", *International Test Conference*, 1993, pp.63-72