An ATPG for Threshold Testing: Obtaining Acceptable Yield in Future Processes

Zhigang Jiang and Sandeep K. Gupta

CENG Technical Report 02-02

Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, California 90089-2562
(213)-740-2251

February 2002

An ATPG for Threshold Testing: Obtaining Acceptable Yield in Future Processes

Zhigang Jiang and Sandeep K. Gupta
Department of EE-Systems
University of Southern California
Los Angeles CA 90089-2562
Tel: (213)740-4474; (213)740-2251
Email: zjiang@poisson.usc.edu; sandeep@poisson.usc.edu

Abstract

As VLSI scaling reaches closer to the laws of physics and to the limits of the fabrication processes, yields, especially at desired speed, will decrease. However, for a large class of applications, chips need not be perfect to be acceptable. In this paper, we describe the notion of threshold testing that can help improve effective yield for future processes. We then develop an ATPG and demonstrate that significant increase in effective yield in test application cost.

1 Introduction

As VLSI fabrication process moves deeper into sub-micron, devices will operate near the limits of physical laws and fabrication will test the limits of abilities of fabrication processes and equipment. Consequently, yields measured in terms of perfect chips, especially at desired speeds, are likely to decrease drastically. This paper develops an approach to enhance yield, for large classes of applications, to mitigate this problem when it becomes important.

A large proportion of digital chips manufactured today are used to implement digitally applications that are essentially analog. One predominant example of this are digital audio and video applications. In these applications, the sound and images are analog when they are captured and are perceived by the end user in essentially analog form. In such digital systems, the original signals captured are not identical to those perceived by the end user due to noise in the input, quantization errors in analog-to-digital and digital-to-analog conversions, lossy compression, and so on.

value. Assume that for each bus, the relative significance (e.g., LSB, MSB, and so on) of each of its constituent bits is known.

Threshold testing of a combinational block of logic can be defined as a type of testing that identifies as acceptable, any fabricated copy of the circuit that contains within the block any modeled fault that, **for any vector**,

(i) does not create an error at any control output, and

(ii) does not create at any output data bus an error whose absolute numerical value exceeds a specified threshold for that bus.

Errors at control outputs can cause data to be routed differently, a different operation to be performed in a subsequent block, to completely change the subsequent computation, and so on. In an overwhelming number of applications, any such change will lead to values at data outputs at this or subsequent clocks to be unacceptably different from that for a fault-free circuit. Hence, we do not consider as acceptable any fault that can cause an error at a control output. However, in some specific applications, this restriction may be relaxed for carefully selected control outputs.

For example, consider a combinational block C_i in a full scan circuit shown in Figure 1. The output of C_i can be viewed as a 3-bit data bus, 2-bit data bus and two control outputs. Assume that in this case, any fault in the target fault list that can not cause any error at control outputs Z_6 and Z_7 and can not create an error of absolute magnitude exceeding 3 and 2 at Bus_1 and Bus_2, respectively, under any of the 2^6 possible vectors at its inputs x_1, ..., x_6, is said to be acceptable.

Note that for simplicity, we can consider all control outputs as another bus with a threshold value of 1. This helps simplify the following discussion. For example, under such naming convention for outputs, the notion of threshold testing can be simplified as follows. Threshold testing of a combinational logic block can be defined as the type of testing that identifies as acceptable, any fabricated copy of a chip that contains within the block any modeled fault that, for any vector, does not create an error at any output bus whose numerical value exceeds the threshold of the bus.

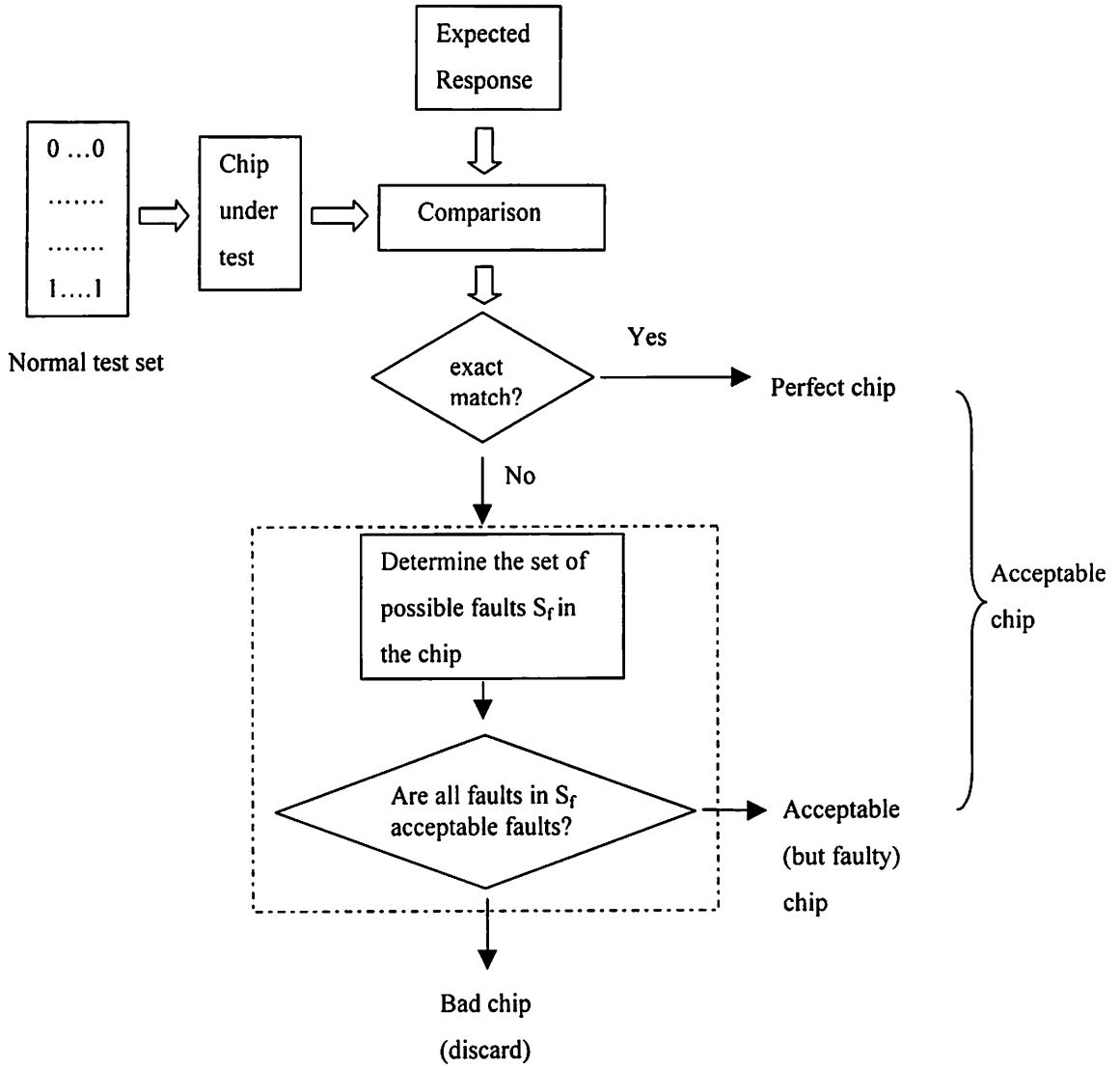

Figure 2. Paradigm 1

3.2 Paradigm 2: Direct Threshold Testing

High complexities of storing and analyzing responses in the diagnostic step of paradigm 1 make its average cost exceedingly high. This is evident in the fact that, in current practice, diagnosis is performed only under special circumstances, such as very low yield. Even then it is applied to a small batch of randomly selected chips [JG 02, Chapter10].

We propose another approach to accomplish threshold testing at a low cost. In this approach, depicted in Figure 3, a *threshold test set* can be used to test each CUT, where a

UC= the cost of applying one vector and analyzing response.

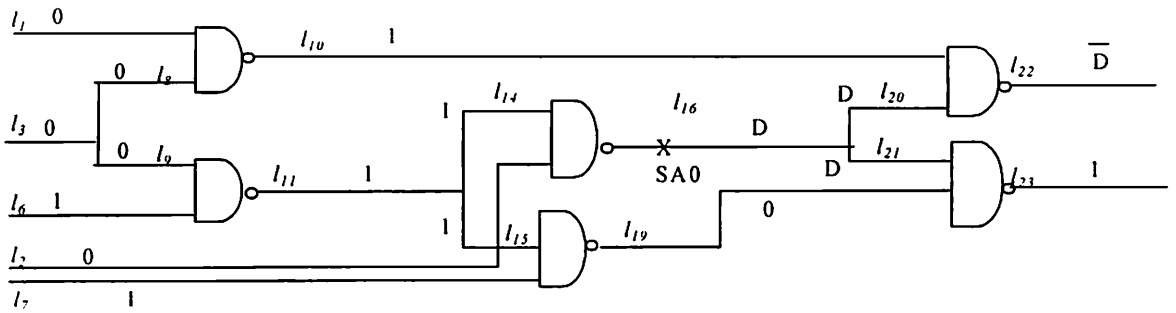

Figure 4. C17 containing fault line 10 stuck at 0 with test vector 00010 applied

So as long as the new test set generated is not prohibitively larger than the normal test set, the cost of paradigm 2 is comparable to that of traditional test methodology.

To put paradigm 2 into practice, an ATPG tool for generating *threshold test vectors* has to be developed, where a *threshold ATPG* tool needs to generate threshold test vectors for all faults in a given fault list.

4 Threshold ATPG

4.1 Terminology

In our study, error threshold is represented by absolute numerical error.

For a data bus with k lines, *absolute numerical error* (ANE) can be defined as:

$$\text{ANE}=\left|\sum_{i=0}^{k-1}(W(i)*(V_{\text{res}}(i)-V_{\text{exp}}(i)))\right|,$$

where $W(i)$ is the weight of ith bit of the bus, $V_{res}(i)$ is the response captured for a particular copy of the chip being tested and $V_{exp}(i)$ is the corresponding response value expected of the fault-free version.

In threshold testing, an error is weighted in terms of the significance of the output at which it appears. A fault in the transitive fan-in of a more significant output tends to cause an error greater than that in the transitive fan-in of a less significant output.

This observation leads us to study characteristics of circuits to see if we can identify acceptable stuck-at faults by merely analyzing circuit topology. If the targeted fault set can be drastically reduced in this means, a normal ATPG can be used with slight modification.

We start with two postulates:

1) If a fault's effect can reach a more significant bit, it will be more likely to be an unacceptable fault.
2) If a fault is located in a more significant arithmetic block, it will be more likely to be an unacceptable fault.

4.2.1 Postulate 1

This postulate is later proved to be wrong in that it doesn't address the phenomenon that adverse fault effects can counteract with each other. So even if the fault effects of a fault can reach outputs with high significance, it doesn't necessarily cause a large numerical error.

A simple example can be found in the adder shown in Figure 6. Consider a stuck-at-1 fault at C_{in}. With the input vector shown in the figure, this fault can have a fault effect at S_3. But the absolute numerical error induced is only 1. And in fact the largest absolute numerical error this fault can cause is 1.

So the ability of a fault's effect to reach more significant bits doesn't directly imply that the fault can cause a large absolute numerical error.

But the converse is true, a fault has to be able to cause an error at bits significant enough to cause a large absolute numerical error. For example, if the threshold is four, a fault has to be able to cause an error at one/more bits more significant than S_1. So the condition that a fault can reach a bit significant enough is a necessary condition but not a sufficient condition.

The efficiency of the proposed ATPG is predicated on the availability of mechanisms to determine an efficient bounding condition.

4.3.1 ANE Directed Branch and Bound

Test generation starts with a completely unspecified vector. Since that implies unknown response at most (nearly all) outputs, the possible ANE it can attain with further specification can be any value between 2^N-1 and 0. As the search for test vector continues, the vector as well as the response gets more specific and possible ANE range narrows in various ways. Three scenarios, shown in Figure 7, may occur for a given desired value of ANE threshold.

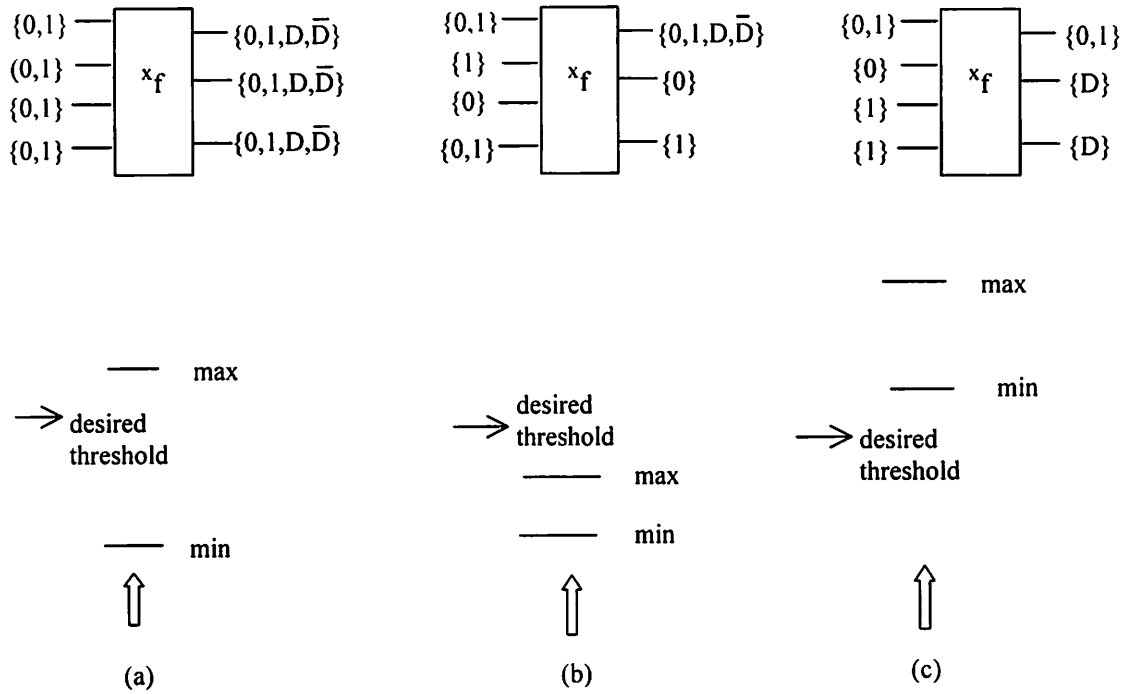

Figure 7. Three scenarios of prospective ANE range

In scenario (a), the desired threshold falls in the possible ANE range. With further specification, an error greater than the threshold may be attained. So the search process should continue branching. In scenario (b), the desired threshold is above the prospective ANE range. With further branching, there is no chance of obtaining an ANE above the desired threshold. In scenario (c), the desired threshold is lower than lower bound on ANE. That means, no matter

A general formula for calculating numerical error for 16-value system is as following:

$$ANE = \sum_{\forall i | V(i) \in \emptyset_S} (0,-1) \bullet W(i) + \sum_{\forall i | V(i) \in \bar{D}_S} (0,1) \bullet W(i) + \sum_{\forall i | V(i) \in \hat{D}_S} (0,-1,1) \bullet W(i)$$

$$+ \sum_{\forall i | V(i) \in \emptyset_S} (-1) \bullet W(i) + \sum_{\forall i | V(i) \in \bar{D}_S} (1) \bullet W(i) + \sum_{\forall i | V(i) \in \hat{D}_S} (-1,1) \bullet W(i) \qquad \text{—— (1)}$$

where

$\emptyset_S$ is the set {{0},{1},{0,1}}, in which no value contains D/$\bar{D}$. D/$\bar{D}$ will not appear at the primary output with further branching.

D_S ={{D}}. A D value currently appears at the primary output.

$\bar{D}_S$ ={{$\bar{D}$}}. A $\bar{D}$ value currently appears at the primary output.

$\hat{D}_S$={{D,$\bar{D}$}}. There definitely will be a D/$\bar{D}$ at the primary output, but one or other may occur depending on the branch taken to further specify the current vector.

D_{PS} is the set{{0,D},{1,D},{0,1,D}}, in which every value contains a D and doesn't contain $\bar{D}$. Note {D} is not a component of D_{PS}. There possibly will be a D at the primary output with further branching.

$\bar{D}_{PS}$={{0,$\bar{D}$},{1,$\bar{D}$},{0,1,$\bar{D}$}}, in which every value contains a $\bar{D}$ and doesn't contain D. Note {$\bar{D}$} is not a component of D_{PS}. There possibly will be a $\bar{D}$ at the primary output with further branching.

$\hat{D}_{PS}$={{0,D,$\bar{D}$},{1,D,$\bar{D}$,}, {0,1,D,$\bar{D}$}}. There possibly will be a D/$\bar{D}$ at the primary output with further branching, but D, $\bar{D}$, or neither may occur depending on further branching.

Note that in Equation (1) to capture ANE, a value such as (0,1) denotes a variable that may become 0 or 1, similarly (0,-1,1) can take values 0,-1, or 1.

	MSB ↓						LSB ↓
Output Bit# (i)	6	5	4	3	2	1	0
Value v(i)	{0, D, $\bar{D}$}	{1,D}	{D}	{1}	{0,1}	{$\bar{D}$}	{$\bar{D}$}

Figure 9. Another example response

4.3.2.3 Lower Bound

The procedure of finding lower bound is a little more involved than that of the upper bound. It also has two phases, namely, finding smallest positive error and finding smallest negative error. The one with smaller absolute value is the lower bound on ANE.

Our algorithm considers four cases separately.

(1) No output has value in D_S, $\overline{D}_S$, or $\hat{D}_S$.

(2) Next three cases assume that at least one output has value in one of the above sets. In each of the following three subcases, let P denote the position of the most significant output that has a value in D_S, $\overline{D}_S$, $\hat{D}_S$.

(a) Value at output P, i.e., V(P)={ D }.

(b) V(P)={ $\overline{D}$ },

(c) V(P)={ $\hat{D}$ }.

For case 1, the lower bound is obviously zero.

For case 2(a), the smallest negative error is obtained by first computing the largest positive error for all outputs less significant than P and adding that to -2^P, the numerical error contribution of output P. Next the smallest positive error can be computed by finding the output Q, that is the output of minimum significance greater than P that has a value in $\hat{D}_{PS}$ or $\overline{D}_{PS}$. A value { $\overline{D}$ }is assigned at Q to contribute 2^Q to numerical error. The greatest negative error is computed for all outputs less significant than Q and added to 2^Q to obtain this minimum positive numerical error. The minimum ANE is obtained by taking the minimum of the absolute values of the smallest positive and negative errors computed above.

Case 2(b) is dual of the above case.

Finally consider the case 2(c). In this case, smallest negative error is computed by assigning {D} at output P and adding to the corresponding numerical error contribution, -2^P, the maximum positive error for bits less significant than P. The smallest positive numerical error is computed by assigning { $\overline{D}$ } to P and adding to the corresponding numerical error contribution, 2^P, the maximum negative error for bits less significant than P. Finally, the minimum ANE is

4.3.3 Threshold ATPG Implementation

We have modified a classical PODEM implementation to use the sixteen valued system described above. We also use our linear complexity algorithm to compute lower and upper bounds on ANE. The values of the lower and upper bounds are used to decide whether PODEM should continue to branch, bound, or stop in the manner depicted in Figure 7.

We also perform equivalence fault collapsing to reduce the number of faults to be targeted by the ATPG.

4.3.4 Example Execution of Proposed Threshold ATPG

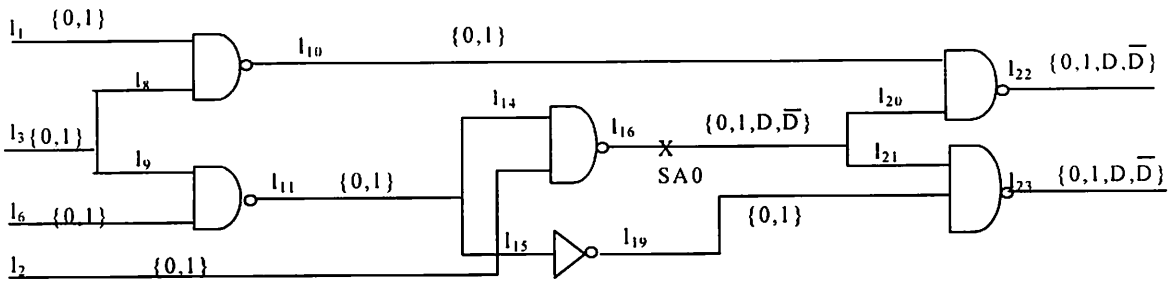

Figure 11. Example circuit with a fault l_{16} SA0

Consider the process of generating a threshold test vector for one fault, (say) l_{16} SA0, for the circuit shown in Figure 11. Assume that the desired threshold is 2, output line l_{22} has a weight of 1 and output line l_{23} has a weight of 2.

Threshold ATPG would basically work with the same fault effect excitation (FEE), fault effect propagation (FEP) subtasks to try to propagate the fault effects to the primary outputs.

Suppose, during FEE, {0} is assigned at l_2, and l_{16} attains a {D}. As shown in Figure 12, the two output lines, l_{22} l_{23}, each attain a $\{1,\overline{D}\}$. Using the error bound computation procedure above, we can find the ANE upper bound is 3, and lower bound is 0. Since desired threshold is between these bounds, we continue branching.

Now D-frontier contains two lines: l_{22} and l_{23}. Suppose l_{22} is chosen for the next FEP subtask and during that task {0} is assigned a l_3. With this assignment, as shown in Figure 13,

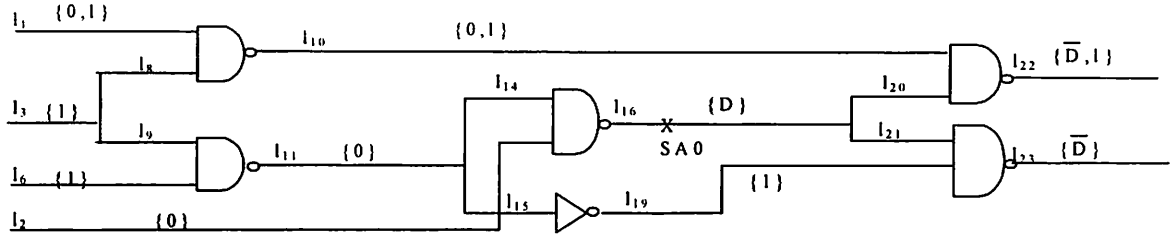

Figure 14. After backtrack, {1} is assigned at l_6 and now the threshold test generation is successful.

5 Experimental Results

The proposed threshold ATPG tool has been implemented. Experimental results are gathered for ISCAS85 benchmark circuits.

5.1 Test set size

In Figure 15, normalized threshold test set size vs. threshold is shown for each ISCAS85 benchmark circuit. In this plot, threshold test set size for each circuit at certain threshold is first normalized with respect to its own classic test set, and then plotted vs. threshold. As can be seen in the figure, when threshold goes as high as 128, threshold test set size for all circuits stays under 1.6 times its classical test set size. In fact for most circuits, the ratio is below 1.2. The average over all circuits of normalized test set size vs. threshold is shown in Figure 16.

These results show that threshold testing can be carried out without any significant increase in test application cost.

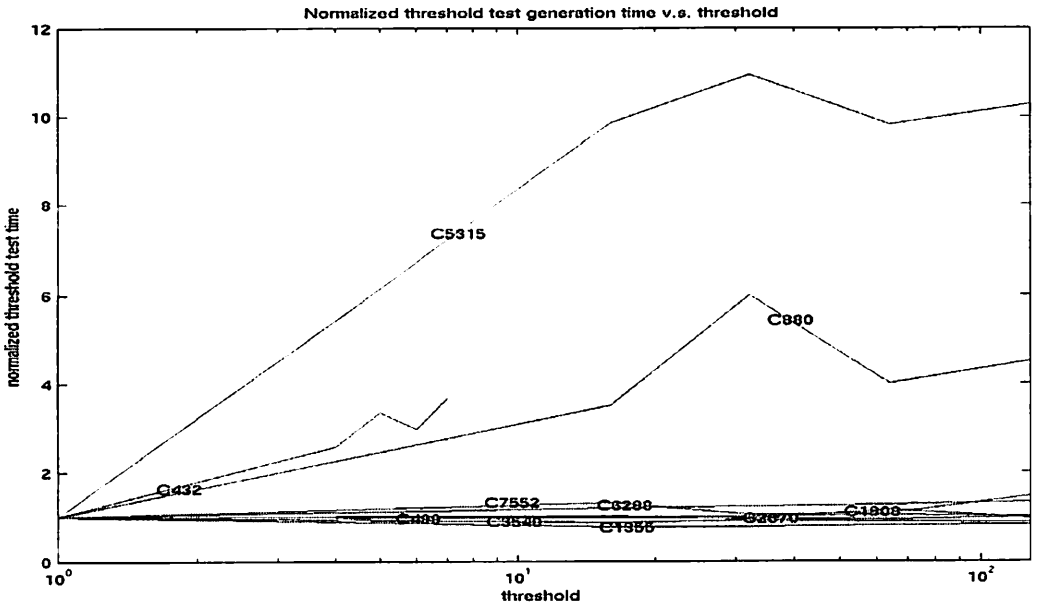

Figure 17. Normalized threshold test ATPG run time vs. threshold

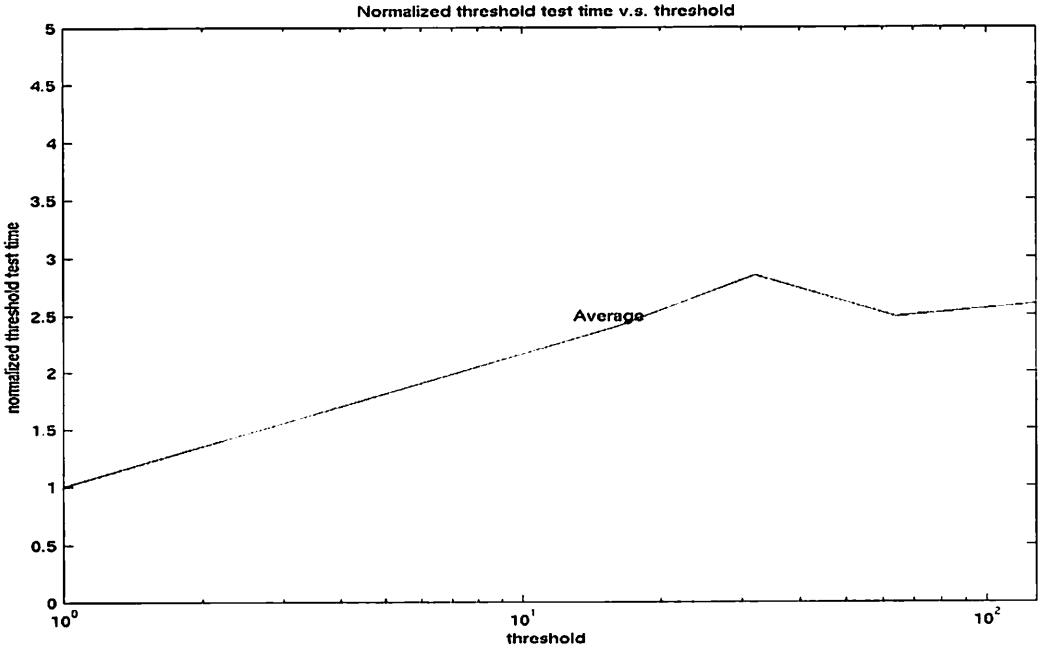

Figure 18. Average of normalized threshold ATPG run time vs. threshold

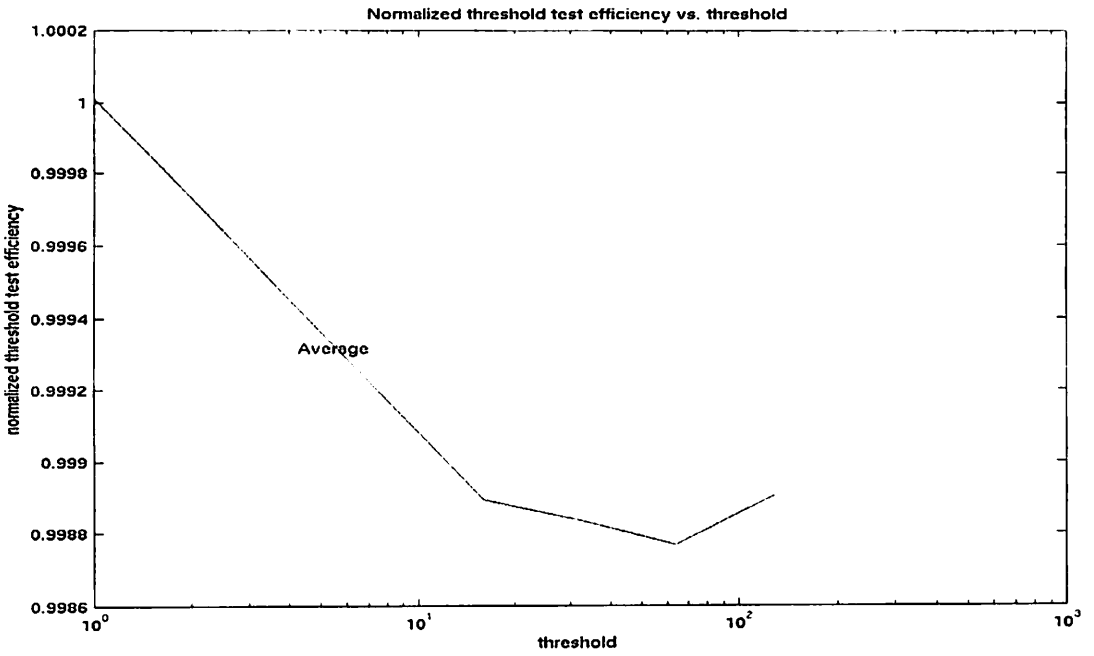

Figure 20. Normalized threshold test efficiency vs. threshold

5.4 Acceptable Faults vs. threshold

Next we analyze the relationship between the threshold value and the percentage of the faults proven to be acceptable at that threshold value. This relationship is important because it determines how the percentage of chips that are imperfect but acceptable grows with desired threshold value.

We limit our examination to the datapath oriented circuits in the ISCAS85 benchmark suite, namely C880, C2670, C3540, C5315, C6288, and C7552. To enable comparison of the trend across different circuits, we use normalized threshold, i.e., threshold value as a percentage of the maximum valued carried by a data bus. Figure 21 shows the percentage of faults proven as acceptable for normalized values of threshold.

Note that the percentage of faults proven acceptable is virtually independent of normalized threshold for circuits like C2670 and C3540 while it increases rapidly with the normalized threshold value for C880, C5315, and C6288. Structural analysis of these circuits explains the difference in this trend. Table 1 shows the percentage of faults in each circuit and

Hence we conclude that for data path dominated circuits, large percentage of faults are acceptable for higher value of the desired threshold. This implies that large percentage of chips can be found to be acceptable despite being imperfect if error threshold can be set higher for an application in which a chip may be used.

5.5 ATPG Time Per Fault

As shown below in Figure 23 for a typical circuit, ATPG time per fault remains virtually constant as threshold is changed, for each type of fault, namely those that are tested (i.e., for which a test is successfully generated), those proven acceptable, and those aborted.

It should be noted that most of the time is consumed by faults that are aborted and those that are proven acceptable. Acceleration of threshold ATPG for these two types of faults is a subject of ongoing research.

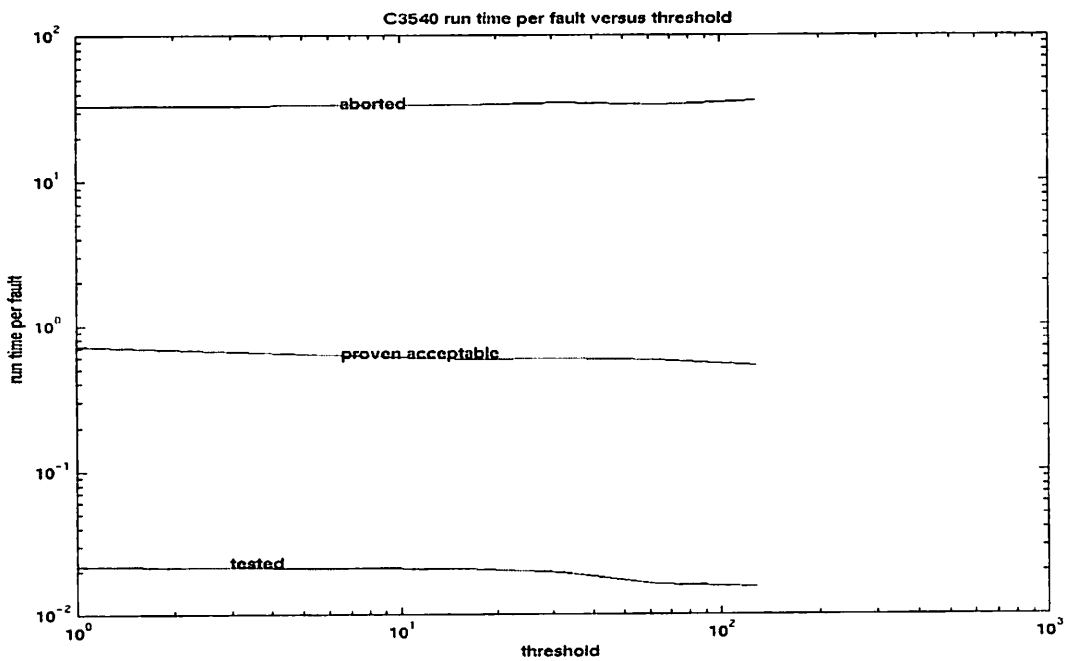

Figure 22. C3540 run time per fault vs. threshold

[KP 94 TCAD] W. Kunz and D. Pradhan, "Recursive Learning: A New Implication Technique for Efficient Solutions to CAD Problems," IEEE Trans. on Computer-Aided Design, 13 (9), pp. 1143-1157