CATAPULT: Concurrent Automatic Testing Allowing Parallelization and Using Limited Topology

Rhonda Kay Gaede
M. Ray Mercer
Kenneth M. Butler
Department of ECE, ENS 143
The University of Texas at Austin
Austin, Texas 78712-1084

Don E. Ross
Microelectronics and Computer
Technology Corporation (MCC)
3500 West Balcones Center Drive
Austin, Texas 78759

Abstract

This paper deals with an improved algorithm for identifying redundant faults and finding tests for "hard faults" in combinational circuits. A new, concurrent approach is proposed which is based upon the concepts of functional decomposition, explicit representation of fanout stems and the Boolean difference. The data structure to be used is the Binary Decision Diagram as developed by Lee, Akers and Bryant. This algorithm operates as a backend to test generators which use random patterns or heuristics or a combination of the two.

1.0 Introduction

Due to the high cost of testing, circuits are being modified in the design process to facilitate testing. One design methodology, level sensitive scan design (LSSD)[Eichelberger 77], has gained increasing acceptance since its introduction in the 70s. This methodology transforms testing from the domain of sequential circuitry into the domain of combinational circuitry only. It is the necessary and growing acceptance of this methodology, and others like it, which has caused great interest in Automatic Test Pattern Generation (ATPG) for purely combinational circuitry. A new algorithm for dealing with the faults left behind by traditional ATPG is the subject of this paper.

1.1 Previous Work

ATPG research for classical stuck-at faults has made many advances in the past twenty years. These developments include single path sensitization [Armstrong 66], the D-algorithm [Roth 66], PODEM (Path Oriented DEcision Making) [Goel 81], FAN [Fujiwara 83], Efficient Handling of Fanout Constraints [Hwang 86], TOPS [Kirkland 87], and SOCRATES (Structure-Oriented Cost-Reducing Automatic TESt pattern generation system) [Schulz 88]. They have all been concerned with reducing the search space necessary to detect faults. They have either made random choices for assignments or based their choices on a testability measure, which by its very nature, makes simplifying assumptions. They are all path-oriented approaches or "path runners", whose characteristics we will now describe.

Path runners manipulate constants at each node in the network. If these constants don't satisfy the current goal, the circuit must be traversed to obtain new data. These path runners expend a limited amount of effort to find a solution for each fault. (A solution consists of either finding a test or identifying a redundancy.) If this amount of effort is exceeded for a fault, they report no solution for it. This amount of effort is typically given in terms of backtracks. The fact that path runners do not achieve cost efficient solutions for hard faults is symptomatic of the mismatch between the tool and the problem under attack.

1.2 Motivation

In essence, the path runners carry along partial information and efficiently find solutions for "easy faults". Our original belief was that a functional approach would be very effective in generating tests for all faults in a circuit. For this to be true, a functional approach would have to perform better in a situation where a test might easily be found with partial information. The data we have gathered has altered our view. As an example, we generated tests for all the single stuck-at faults in the 74LS181 ALU circuit and this process required approximately one CPU minute to complete on a SUN3 system. This performance does not compare favorably with test generation times which have been previously reported. Therefore, our focus shifted to a harder problem which was better matched to our tool.

If the problem is difficult, we need complete information in order to obtain the solution. For path runners to obtain complete information would require serial exhaustion of a very large search space. Our approach is different from path runners in that we manipulate functions rather than constants and, accordingly, less of our processing is associated with the topology of the circuit. Path runners perform the same task at one node multiple times if they are searching a large space while we perform each task once and save the result. It is the faults which actually require a good deal of search which are addressed by our algorithm. This is the area in which a functional approach is a tool which is well matched to its problem.The algorithm we are proposing here gathers exact information in a concurrent manner. Information which is not fault specific is propagated through the circuit.This information can be used while processing all of the remaining faults.

The remainder of this paper is devoted to the details of the algorithm and the results of our investigations. The first section deals with the concurrent nature of test as we use it. This section is followed by a section on our data structure, BDDs. Having these basics, the algorithm is explained in detail.The results of the implementation of the algorithm are given next. Finally, the significance of this research is discussed.

2.0 Concurrent Test Generation

In test generation, there are two requirements that must be met to find a test. The fault site must be controllable from the primary inputs and the fault indication must be observable at a primary output. Control information lends itself well to a concurrent approach. The controllability function of a gate output can always be calculated from the controllability functions of the inputs to the gate. This fact allows all controllability functions to be calculated *in one pass from the primary inputs to the primary outputs*. Unfortunately, the observability of a fanout stem cannot generally be calculated from the observabilities of its branches. In order to overcome this problem, we propose representing the *output functions in terms of fanout stems as well as primary inputs*. By doing this, the observabilities of the fanout stems can be calculated by using the Boolean difference. With this representation, the observability information can be obtained in one pass from the represented stems to the fault sites.

This concurrent approach demands a data structure which

This work was supported by the National Science Foundation under Grant MIP-8552537, MCC and by the AT&T Foundation together with the AT&T Information Systems Laboratories.

makes efficient use of memory in representing the controllability and observability functions and which requires a reasonable time complexity to perform logical operations on them. We believe the best candidate for this data structure to be that of Binary Decision Diagrams (BDDs) as they have been developed over the past half-century. [Shannon 38] [Lee 59] [Akers 78] [Bryant 86].

2.1 Binary Decision Diagrams

Previous canonical representations of circuits or switching functions such as truth tables, Boolean equations and Karnaugh maps all have the unpleasant property of growing exponentially with the number of variables involved. A BDD is a switching function representation which is reported to be exponential in the number of variables only in rare cases (such as an integer multiplier) [Bryant 86].

An example of a BDD is given in Figure 1. The switching function represented is $f(A,B,C) = A + BC$. Nodes are labelled with switching variables and the output arcs are labelled with values of associated switching variables. Terminal nodes may also have a value of X in addition to the values of 0 and 1 which are shown here. Each path in a BDD to a 1 represents an implicant of the switching function. Since a BDD has an input ordering associated with it, these paths may or may not be prime implicants.

A BDD is a graphical representation of Shannon's decomposition. For example, the subgraph for A=1 corresponds to $f(1,B,C) = 1$ and the subgraph for A=0 corresponds to $f(0,B,C) = BC$. This representation is both canonical and minimized. The complexities of storing and manipulating BDDs are given in [Bryant 86].

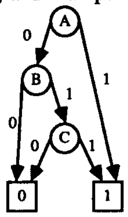

Figure 1. BDD example

2.2 Fault Set

Currently, our system accepts descriptions of combinational circuits expressed as interconnections of logic gates from the set {BUFFER, NOT, AND,NAND,OR,NOR, XOR}. Within this description, the fault model is currently restricted to the classical stuck-at fault at a single gate lead. It has been shown [Breuer] that selection of the fault pairs s-a-1 and s-a-0 on each of the primary inputs and each of the fanout branches, along with the output of each XOR gate, will create a fault set which is representative of all faults.

This fault set, called the checkpoint set, is not completely minimized with respect to equivalence relations, and could be further reduced and still be representative of all faults. Also, because the checkpoint set exploits dominance relations, any fault represented by a redundant fault through a dominance relation requires extra processing to determine whether it is redundant as well, because it may be testable [Abramovici 86]. For this reason, use of the checkpoint fault set can cause coverage statistics to be slightly inaccurate unless checkpoint faults can be proven redundant and tests can be generated for the non-redundant faults represented through a dominance relation by a redundant fault. Despite slight inaccuracies in reported coverage of checkpoint sets, this set is usually chosen for large designs due to its small size relative to previously popular fault sets, such as the one obtained by exploiting only equivalence relations.

The "path runner" against which we are comparing our algorithm, TOPS [Kirkland 87], follows random test pattern generation with very fast fault simulation to cover most of the faults [Waicukauski 85]. This technique obtains a fault coverage of more than 90% with very little effort and then attacks the set of faults left behind after random test pattern generation. We arrive at our target fault set by running TOPS with the backtrack limit set to 10 and using the faults which require more than 10 backtracks as our target set.

2.3 Algorithm Overview

At this point, it's germane to discuss the general format of the algorithm at a high level before descending into a detailed description. The first step in the algorithm is to compute controllabilities in one pass from the primary inputs to the primary outputs using BDDs as the data structure. Controllability is defined in our algorithm as the switching function at some point in the circuit. During this pass, decomposition occurs at a subset of the fanout stems in the circuit, i.e. the fanout stems are explicitly represented as nodes in the BDDs. The subset of the fanout stems used is the set of fanout stems necessary to determine the observabilities of the fault sites. At the primary outputs, the functions are in terms of the primary inputs and the fanout stems represented. From these functions, by use of the Boolean difference and composition, the observability of each represented fanout stem is determined. Observability is defined in our algorithm to be the switching function which allows a change at some point in the circuit to affect at least one primary output. Fault site observabilities are then determined from fanout stem observabilities by using backward propagation of observabilities.

Having both the controllability and the observability information at a fault site during the backward pass, the two BDDs which represent these quantities are ANDed together, yielding a test BDD for the fault. It now remains to find a test in terms of primary inputs only. We do this by partially composing test BDD fanout stem functions along a path until a test is found or the path fails. When a path fails, partial composition occurs along another path. This process continues until a test is found or the space is exhausted and the fault is declared to be redundant. The idea of "partial composition" is a new one which we will explain in detail in a later section.

3.0 Algorithm Description

It is useful at this point to define the terminology which we will be using throughout this discussion. A circuit can be generally characterized by the terms in Figure 2.

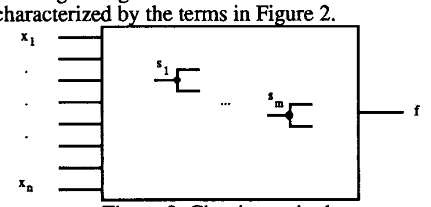

Figure 2. Circuit terminology

The circuit has n primary inputs, m represented fanout stems and one output. The m fanout stems are ordered in the following way. Stem s_1 is as close to or closer to the primary inputs than s_2 is. Initially, we will deal with a single output and later talk about extension to multiple outputs.

3.1 Calculating Controllability Information

We have previously mentioned the idea of explicitly representing functions in terms of fanout stems as well as primary inputs. This method of representation is closely related to the concept of functional decomposition. During the forward pass to calculate controllability information, we find

$$s_1(x_1,\ldots,x_n)$$
$$s_2(x_1,\ldots,x_n,s_1)$$
$$\vdots$$
$$s_m(x_1,\ldots,x_n,s_1,\ldots,s_{m-1})$$
$$f(x_1,\ldots,x_n,s_1,\ldots,s_{m-1},s_m) \qquad \text{(notation from [Hwang 86].)}$$

3.2 Calculating Observability Information

Having calculated the controllability information, we now proceed to the calculation of the observability information. The observability of s_m is determined first. We get this by using the Boolean difference [Akers 59].

$$df/ds_m = f(x_1,\dots,x_n,s_1,\dots,s_{m-1},0) \oplus f(x_1,\dots,x_n,s_1,\dots,s_{m-1},1)$$

Once we have the observability of s_m, we can continue by finding the observability of s_{m-1}. In order to find the observability of s_{m-1}, we must express the function, f, in terms of $s_1,s_2,\dots,s_{m-1}$ and remove the dependence on s_m. So, we compose the function, f, using the function s_m. Composition can be expressed in terms of restriction and Boolean operations according to the following expansion derived directly from Shannon's expansion theorem.

$$f_1|_{x_i=f_2} = f_2 \bullet f_1|_{x_i=1} + (\neg f_2) \bullet f_1|_{x_i=0}$$

$$f(x_1,\dots,x_n,s_1,\dots,s_{m-1}) = s_m(x_1,\dots,x_n,s_1,\dots,s_{m-1}) \bullet f(x_1,\dots,x_n,s_1,\dots,s_{m-1},1)$$
$$+(\neg s_m(x_1,\dots,x_n,s_1,\dots,s_{m-1}))\bullet f(x_1,\dots,x_n,s_1,\dots,s_{m-1},0)$$

After the composition of s_m, we use the Boolean difference again to calculate df/ds_{m-1}. These processes are continued until the last stem observability is found

$$df/ds_1 = f(x_1,\dots,x_n,0) \oplus f(x_1,\dots,x_n,1)$$

and f is determined in terms of primary inputs only.

$$f(x_1,\dots,x_n) = s_1(x_1,\dots,x_p)\bullet f(x_1,\dots,x_n,1) + \neg\ s_1(x_1,\dots,x_n)\bullet f(x_1,\dots,x_n,0)$$

At the completion of these processes, we have determined all of the stem observabilities for only one output. The observabilities can be extended to include observabilities at multiple outputs such as occur in the circuit of Figure 3. Take stem s_m, for example. The observability of s_m with respect to each output can be computed and the individual quantities ORed together in order to obtain the total observability. That is, $dF/ds_m = df_1/ds_m + df_2/ds_m + \dots + df_p/ds_m$. The total stem observabilities can now be used to find the total observabilities of the fault sites.

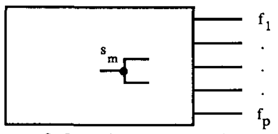

Figure 3. Multiple output observability

The mechanism which allows us to find a fault site's observability is the backward propagation of observability information from a stem. As a simple example of this mechanism, consider an AND gate.Given the fact that the observability of the output is known, what does it take to calculate the observability of an input? From the fact that the gate is an AND gate, we know that all inputs other than the input of interest must be a one in order for the input of interest to be observable. Requiring a one on a wire is equivalent to the one function controllability of a wire. So, the observability of the input of interest is the observability of the output ANDed with the controllabilities of the other inputs.

3.3 Obtaining Test Diagrams

At this point, we have made a forward controllability pass and a backward observability pass. It now remains to obtain the BDD of the test for a particular fault. Since we have the two quantities necessary, it is a simple matter to find the test BDD. All that is required is to AND the controllability BDD and the observability BDD of a wire if the fault in question is s-a-0, or AND the inverse of the controllability BDD and the observability BDD if the fault is s-a-1.

3.4 Obtaining a Test in Terms of Primary Inputs Only

We now have the test BDD in terms of fanout stems and primary inputs. Ultimately, we must have the test for a fault expressed in terms of primary inputs only because they are the sole points of access to a circuit under test. A path to the 1 terminus in a test BDD represents a test. If there is such a path in the test BDD which does not pass through any of the nodes $s_1,\dots,s_m$, we have a test for the fault in terms of primary inputs only and no further action is required. However, this case will not occur very often and, in most cases, we will have to do some composition work. This composition can be terminated whenever a path to the terminus 1 through primary input nodes only occurs. The composition work we do is composition along paths or "partial composition".

In order to explain partial composition, it's timely here to talk about composition in graphical terms. Composition can be thought of as expanding a node in the graph into the nodes of the function it represents. We will call the expansion of every node of a variable, say s_2, in a test BDD "complete composition". Complete composition is expressed by the composition equation given earlier and repeated here.

$$f(x_1,\dots,x_n,s_1,\dots,s_{m-1}) = s_m(x_1,\dots,x_n,s_1,\dots,s_{m-1}) \bullet f(x_1,\dots,x_n,s_1,\dots,s_{m-1},1)$$
$$+(\neg s_m(x_1,\dots,x_n,s_1,\dots,s_{m-1}))\bullet f(x_1,\dots,x_n,s_1,\dots,s_{m-1},0)$$

However, it is possible to do less than complete composition. By this, we mean that only selected nodes of a variable might be expanded. The following discussion deals with the example test BDD of Figure 4. As can be seen, there are two s_2 nodes in this test BDD. It is valid and desirable to expand only one of these nodes in order to keep the size of the resultant BDD small. In order to understand expansion of only one s_2 node in a mathematical sense, we must consider the function to be a sum of implicants. Each of these implicants is itself a function and can be composed individually. So, implicants which contain s_2^* are composed while implicants which contain s_2 are not.

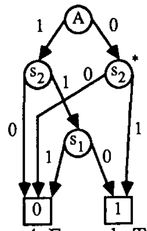

Figure 4. Example Test BDD

If, at any time during these expansions, a path is found to the terminus 1 through primary input nodes only, no further expansion is necessary. We select the node to expand by selecting a path to the terminus 1 in the test BDD. A path to the terminus 1 in a test BDD represented in terms of both fanout stem nodes and primary input nodes is considered a possible test. It may be that a condition required by a fanout stem node, which is assumed to be independent, conflicts with a condition required on a primary input and that this fact is masked by the assumption of independence of the fanout stem node in the decomposed representation.

Therefore, when we compose along a path, we either find a conflict, in which case the subgraph we were expanding collapses to 0 or we find a test. It is this feature of the subgraph collapsing to 0 which keeps the size of the BDD under control. If one path fails, we begin to compose along another path. If all paths fail, the entire graph collapses to the terminus 0. In the case of the example test BDD, we can see that there are two paths to the terminus 1. We choose to compose s_2^* first. The function represented by the example test BDD is

$$f(A,B,s_2,s_1) = I_1(A,B,s_2,s_1) + I_2(A,B,s_2,s_1) = As_2s_1' + A's_2^*,$$

where I_1 and I_2 are implicants of f.

Given that $s_2 = B' + s_1'$,composing only s_2^* gives

$$f = I_1(A,B,s_2,s_1) + s_2' \bullet I_2(A,B,0,s_1) + s_2 \bullet I_2(A,B,1,s_1)$$
$$= As_2s_1' + (Bs_1)0 + (B' + s_1')A'$$

Then, $f = As_2s_1' + A'B' + A's_1'$

This partial composition can be done graphically by using a slight modification of the compose algorithm given in [Bryant 86]. The result of the partial composition on the BDD of Figure 4 is given in Figure 5. At this point, a test has been found, A=0 and B=0. In general, further compositions may be required. After each composition, the paths available must be reevaluated as the paths may change with every composition.The representation in terms of Boolean algebra can be easily shown to be equivalent to the BDD representation.

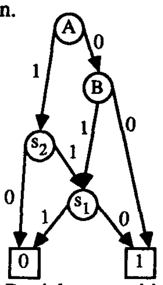

Figure 5. Partial composition result

4.0 Results

We have implemented the algorithm described above in the C language. We are currently running it on the same system upon which TOPS is running at MCC, a SUN4.Of the circuits which have been established as benchmarks for combinational ATPG[Brglez 85], there are only three which have hard faults. To date only one of these has been investigated, the c432. As might be imagined, ours is a memory intensive approach. To provide some idea of the magnitude of the memory requirement, there are two numbers given in Table 1 for the circuits we have investigated. The first number is the size, in nodes, of the largest BDD encountered while processing the circuit. The second number is the average BDD size for all the BDDs used during the processing.

Circuit	max BDD size	avg. BDD size
c432	524	141

Table 1. BDD size data

Even though this is a memory intensive process, it still has good performance characteristics for the case we investigated. This is true in spite of the fact that no clever ways of dealing with the memory have been implemented. In fact, all of the controllability BDDs are being retained when only a subset of them is required. In Table 2, we compare the performance of our algorithm with that of TOPS for the target fault set we have discussed. The timing data is given per circuit for both TOPS and CATAPULT measured in seconds.. Also given is the number of backtracks performed for each fault by TOPS. These numbers indicate that the total time number is not being dominated by the time required to process a single fault. Our algorithm performs significantly better than TOPS in this domain for the c432.

Circuit	Fault site destination	Fault site source	Fault stuck value	Number of backtracks	Time (TOPS)	Time (CATAPULT)
c432	379GAT	360GAT	0	52,656	311	11
	347GAT	319GAT	0	19,664		
	259GAT	213GAT	0	27,063		
	379GAT	115GAT	0	52,656		
	347GAT	112GAT	0	19,664		
	259GAT	102GAT	0	27,063		

Table 2. Timing performance

5.0 Conclusion

This paper presents an exciting new approach to combinational ATPG in a concurrent manner. This approach focuses on hard faults which are not effectively handled by path runners. We present the use of a novel data structure, that of BDDs, in a way that overcomes the problem of calculating observabilities in a single backward pass through the circuit. Although this is a memory intensive approach, its promise is demonstrated by preliminary results.

Acknowledgements

We would like to thank MCC for the use of their computer on which the data was collected. We would also like to thank MCC and Joe Newman for his implementation of TOPS in C.

References

[Akers 59]Sheldon B. Akers Jr.,"On a Theory of Boolean Functions", Journal of the Society of Industrial and applied Mathematics, No. 4,December 1959,pp. 487-498.

[Akers 78]Sheldon B. Akers,Jr.,"Binary Decision Diagrams", IEEE Transactions on Computers,Vol. C-27,No. 6, June 1978, pp. 509-516.

[Abramovici 86] M. Abramovici, P. R. Menon and D. T. Miller, "Checkpoint Faults are not Sufficient Target Faults for Test Generation", IEEE Transactions on Computers, Vol. C-35, No. 8, August 1986, pp. 769-771.

[Armstrong 66] D. B. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets", IEEE Transactions on Electronic Computers, Vol. EC-15, Feb. 1966, pp.66-73.

[Breuer 76]M.A. Breuer and A.D. Friedman, Diagnosis & Reliable Design of Digital Systems,Computer Science Press, 1976.

[Brglez 85]F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran",Proceedings of the IEEE International Symposium on Circuits and Systems, June 1985.

[Bryant 86] Randal E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers, Volume C-35, No. 8, August 1986, pp. 677-691.

[Eichelberger 77] E. B. Eichelberger and T. W. Williams, " A Logic Design Structure for LSI Testability ", 14th Design Automation Conference Proceedings, 1977, pp. 462-467.

[Fujiwara 83] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms", IEEE Transactions on Computers, Vol. C-32, No. 12, December 1983.

[Goel 81] Prabhakar Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE Transactions on Computers, Vol. C-30, No. 3, March 1981, pp. 215-222.

[Hwang 86] Ki Soo Hwang and M. Ray Mercer, "Derivation and Refinement of Fan-Out Constraints to Generate Tests in Combinational Logic Circuits", IEEE Transactions on Computer-Aided Design, Vol. CAD-5, No. 4, October 1986, pp. 564-572.

[Kirkland 87] Tom Kirkland and M. Ray Mercer, "A Topological Search Algorithm for ATPG", Proceedings of the 24th ACM/IEEE Design Automation Conference,June 1987, pp. 502-508.

[Lee 59] C. Y. Lee, "Representation of Switching Circuits by Binary-Decision Programs", Bell System Technical Journal, Vol. 38, July 1959, pp. 985-999.

[Roth 66] J. Paul Roth, "Diagnosis of Automata Failures: A Calculus and a Method", IBM Journal of Research and Development, Vol. 10, July 1966, pp. 278-291.

[Schulz 88] M. H. Schulz, E. Trischler and T. M. Sarfert, IEEE Transactions on Computer-Aided Design, Vol. CAD-7, No. 1, January 1988, pp. 126-137.

[Shannon 38] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits", Transactions of the AIEE, Vol. 57, 1938, pp. 713-723.

[Waicukauski 85]J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom and T. McCarthy, "Fault Simulation for Structured VLSI", VLSI Systems Design, December 1985, pp. 20-32.