An Automatic Test Pattern Generation Framework for Combinational Threshold Logic Networks

Pallav Gupta, Rui Zhang, and Niraj K. Jha
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544
{pgupta|rzhang|jha}@ee.princeton.edu

***Abstract*— We propose an automatic test pattern generation (ATPG) framework for combinational threshold networks. The motivation behind this work lies in the fact that many emerging nanotechnologies, such as resonant tunneling diodes (RTDs) and quantum cellular automata (QCA), implement threshold logic. Consequently, there is a need to develop an ATPG methodology for this type of logic. We have built the first automatic test pattern generator and fault simulator for threshold logic which has been integrated on top of an existing computer-aided design (CAD) tool. These exploit new fault collapsing techniques we have developed for threshold networks. We perform fault modeling to show that many cuts and shorts in RTD-based threshold gates are equivalent to stuck-at faults at the inputs and output of the gate. Experimental results with the MCNC benchmarks indicate that test vectors were found for all testable stuck-at faults in their threshold network implementations.**

I. Introduction

Although complementary metal-oxide semiconductor (CMOS) technology is not predicted to reach fundamental scaling limits for another decade [1], alternative emerging technologies are being researched in hopes of launching a new era in nanoelectronics. One of the most promising nanotechnologies for mid-term, post-CMOS technology is RTDs coupled with heterostructure field-effect transistors (HFETs) [2]. This is due to several reasons. First, RTDs can be grown with great precision and uniformity using molecular-beam epitaxy. Second, RTD-HFET networks have been demonstrated to work at very high clock frequencies [3]. Finally, RTD-HFETs implement threshold logic which provides improved computational functionality through smaller network logic depth, fewer devices, and shorter wiring [4].

A tool called TELS [5] has been developed recently to synthesize combinational threshold networks. Thus, the next step in a design flow for RTDs is an ATPG methodology for threshold networks. There has been no work done to date in this area that the authors are aware of. Since RTDs are in their infancy and because a full understanding of the behavior and properties of RTDs is currently lacking, we must develop an ATPG framework at the logic level first. This guarantees that our methodology will be applicable to threshold networks, in general, and will be independent (for the most part, except the fault model) of technology mapping – the nanotechnology chosen to implement threshold networks.

The purpose of this paper is to present an ATPG framework for combinational threshold networks. The novel contributions of this work are as follows:

- This is the first ATPG methodology for combinational threshold networks built upon sound mathematical principles and existing ideas in Boolean testing.
- We have implemented our framework on top of a logic synthesis tool, namely SIS [6], to create the first threshold network fault simulator and automatic test pattern generator.

The remainder of this paper is organized as follows. Section II presents background material that is required to understand the ideas presented in this paper and describes previous work. Section III outlines the questions that need to be addressed in our ATPG methodology for combinational threshold logic networks. The theory needed to develop such a methodology is described in detail in Section IV. The methodology is presented in Section V and the experimental results are discussed in Section VI. We conclude the paper in Section VII.

II. Preliminaries and Previous Work

In this section, we present some preliminary concepts to help the reader understand the remainder of the paper better.

A. Threshold Functions

A linear threshold function is a multi-input function in which each digital input, x_i, $i \in \{1, 2, \ldots, n\}$, is assigned a weight w_i such that the output function assumes the value 1 if and only if the weighted sum of the inputs equals or exceeds the value of the function's threshold, T [7]. That is,

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq T + \delta_{on} \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i < T - \delta_{off}. \end{cases} \quad (1)$$

Parameters δ_{on} and δ_{off} are positive numbers that represent defect tolerances that may be considered since variations in the weights (due to manufacturing defects, temperature changes, etc.) can lead to network malfunction. When defect tolerance is not considered, δ_{on} and δ_{off} can be set to 0. A linear threshold gate (LTG) is a multi-terminal device that implements a threshold function. We will use the weight-threshold vector $\langle w_1, w_2, \ldots, w_n; T \rangle$ to denote the weights and threshold of a threshold gate.

B. Realizing a Threshold Function

A threshold function can be realized by a monostable-bistable transition element (MOBILE) such as the one shown in Fig. 1(a) [2]. Fig. 1(b) shows a MOBILE's equivalent LTG representation. The modulation current, ΔI, applied at the output node determines what digital state the device transitions to [4]. The modulation current is obtained from Kirchoff's Current Law and is given as,

$$\Delta I = \sum_{i=1}^{N_p} w_i I(V_{gs}) - \sum_{i=1}^{N_n} w_i I(V_{gs}), \quad (2)$$

Acknowledgments: This work was supported by NSF under Grant No. CCR-0303789.

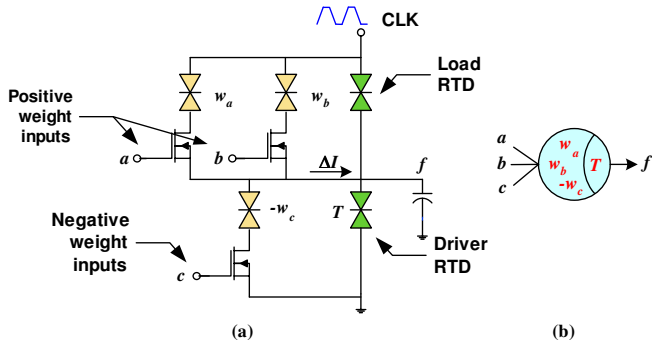

Fig. 1. (a) An LTG implemented using a MOBILE, and (b) its schematic representation.

where N_p and N_n are the number of positive and negative weighted inputs, respectively, and $I(V_{gs})$ is the peak current of a minimum-sized RTD. The net RTD current for the load and driver is $I_T = TI(V_{gs})$. Consequently, the output is logic high if $\Delta I - I_T$ is positive and logic low otherwise.

C. Previous Work

Research in Boolean testing has flourished since the 1960s [8]. On the other hand, there has been virtually no research in testing of arbitrary threshold networks. The bulk of research in threshold logic was done in 1950s and 1960s and focused primarily on the synthesis of threshold networks [7]. Recently, a practical methodology for synthesis of multi-level threshold networks has been presented [5]. Finally, a survey of technologies capable of implementing threshold logic can be found in [9].

III. Questions to be Addressed

Some of the interesting questions that need to be addressed in ATPG for combinational threshold networks are as follows:

1) How is a fault modeled in an LTG?
2) Are test generation and redundant faults as intricately related in threshold ATPG as they are in Boolean ATPG?
3) What condition must be satisfied to excite a fault?
4) How are the propagation D-cubes and singular covers of a threshold function determined?
5) Do any universal relationships exist between the inputs and output of an LTG that we can exploit to reduce the number of faults in the faults list, i.e., perform fault collapsing?

We answer these questions in the next section.

IV. ATPG for Combinational Threshold Networks

In this section, the theory that is needed to develop an ATPG methodology for combinational threshold networks is presented. We begin with fault modeling. We then relate redundant faults to threshold ATPG. Next, we derive the conditions that need to be satisfied in order to find a test vector for a given fault. We show one possible way to determine the propagation D-cubes and singular covers of a threshold function. Finally, we develop some theorems for fault collapsing.

A. Fault Model

Before developing a fault model, it is important to pinpoint the defects that are most likely to occur in RTDs. Given their integration with HFETs, cuts and shorts are a certainty. Furthermore, RTD area variation will cause peak current fluctuations which will cause logic faults [10]. While it is difficult to model and detect parametric defects, in general, it is possible to model them if we make the assumption that the weight will change in such a way that the implemented threshold function will be different from the original threshold function. Due to limited space, we will not address parametric faults here.

Fig. 2(a) shows the cuts and shorts in a MOBILE gate that can be modeled as single stuck-at faults (SSFs) at the logic level. A cut (sites 1, 2, and 3) on an HFET or on a line connecting the RTD and HFET will render it permanently non-conducting and is modeled as a stuck-at-0 (SA0) fault. Similarly, a short across an RTD (site 4) or the driver RTD (site 8) is also modeled as an SA0 fault because in the former, the input weight will become zero while in the latter, there will be a direct connection between the output and ground. A cut at site 6 represents either an SA0 or SA1 fault depending upon the threshold of the gate. If the threshold is less than zero, then the cut is modeled as an SA1 fault. Otherwise, it is modeled as an SA0 fault. On the other hand, faults at sites 5 and 7 are modeled as SA1 faults. A short across the HFET will make it conduct permanently while a direct connection between the output and bias voltage will exist in the presence of a short across the load RTD, making the fault appear as an SA1 when the MOBILE gate is active. These fault models have been verified through HSPICE simulations. HSPICE models for RTD-HFET gates were obtained from [10].

B. Irredundant Threshold Networks and Redundancy Removal

In Boolean testing, irredundant networks are intricately related to circuit testability [8]. We define similar concepts for irredundant threshold networks and threshold ATPG.

Definition 1: A combinational threshold network, G, is irredundant if the removal of any node or edge in G does not represent the same set of Boolean functions.

Definition 2: If no test vector exists for fault s in G, then s is redundant.

This leads us to the following theorem that relates redundancy and ATPG for threshold networks.

Theorem 1: Given network G, if no test vector exists to detect fault s, then the corresponding node or edge in the network can be removed without affecting the functionality of G.

The rules for removing a redundant fault in a threshold network are as follows:

1) If an SA0 fault on an edge is redundant, the edge in the network can be removed, as shown in Fig. 2(c).
2) If an SA1 fault on an edge is redundant, the edge in the network can be removed, as shown in Fig. 2(d). The threshold of the nodes in the edge's fanout must be lowered by the weight of the removed edge.

Furthermore, all nodes and edges in the sub-network that do not fan out and are in the transitive fanin of the removed edge can be removed from the network in both cases.

C. Test Generation

We must now find a test vector for a given fault in an LTG. To find a test vector for a fault at x_i, it is necessary that the term $w_i x_i$ in Equation (1) be the dictating factor in determining the output value of the function.

Theorem 2: Given an LTG implementing the threshold function, $f(x_1, x_2, \ldots, x_n)$, to find test vectors for x_i SA0 and x_i SA1, we must find an assignment on the remaining input variables such that

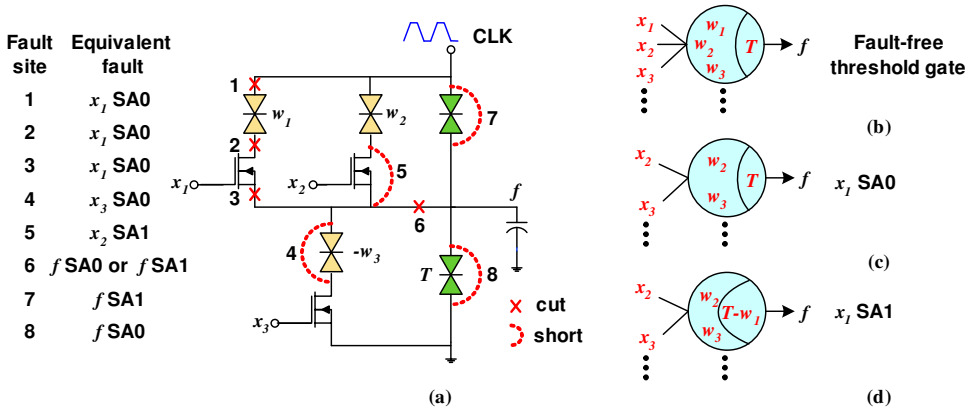

Fig. 2. (a) Fault modeling and fault simulation of an LTG with (b) no faults, (c) an SA0 fault, and (d) an SA1 fault.

one of the following inequalities is satisfied:

$$T + \delta_{on} - w_i \leq \sum_{j=1, j\neq i}^{n} w_j x_j < T - \delta_{off} \tag{3a}$$

or

$$T + \delta_{on} \leq \sum_{j=1, j\neq i}^{n} w_j x_j < T - \delta_{off} - w_i. \tag{3b}$$

If an assignment exists, then $\langle x_1, x_2, \ldots, x_i = 1, \ldots, x_n \rangle$ and $\langle x_1, x_2, \ldots, x_i = 0, \ldots, x_n \rangle$ are test vectors for x_i SA0 and x_i SA1 faults, respectively. If no assignment exists, then *both* faults are untestable and, therefore, redundant.

Theorem 3: Given an LTG implementing the threshold function, $f(x_1, x_2, \ldots, x_n)$, if there exist two (or more) inputs x_j and x_k such that $|w_j| = |w_k|$, then test vectors to detect x_k SA0 and x_k SA1 can be obtained simply by interchanging the bit positions of x_j and x_k in the SA0 and SA1 test vectors for x_j respectively, assuming they exist.

The properties can best be demonstrated by an example. Consider an LTG that realizes the threshold function, $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3$, with weight-threshold vector $\langle 2, 1, 1; 3 \rangle$. For simplicity, assume $\delta_{on} = \delta_{off} = 0$. To test for x_1 SA0, the inequalities to be satisfied are $1 \leq \sum_{j=2}^{3} w_j x_j < 3$ or $3 \leq \sum_{j=2}^{3} w_j x_j < 1$. This leads to three test vectors namely, 101, 110, and 111. The test vectors for x_1 SA1 can be easily obtained by replacing $x_1 = 1$ with $x_1 = 0$ in the original test vectors. Thus, vectors 001, 010, and 011 detect x_1 SA1. Finally, given that vector 110 is a test for x_2 SA0 and since $w_2 = w_3$, a test vector that detects x_3 SA0 is obtained by interchanging the bit positions of x_2 and x_3 in 110 to get 101.

D. Propagation D-cubes and Singular Covers

Propagation D-cubes are used in D-algorithm to sensitize a path from the fault site to one (or more) primary outputs. Knowing the threshold function that is implemented by an LTG, we can resort to algebraic substitution to determine the propagation D-cubes by using the D-notation [8]. For example, to determine the propagation D-cubes of x_1 SA1 in $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3$, substituting D for x_1 in f we get $Dx_2 + Dx_3$. For the fault to propagate, it is required that only the cubes containing D (or $\bar{D}$) get "activated" in f. In this case, since both cubes contain D, activating either or both cubes will result in a propagation D-cube. Thus, the propagation D-cubes for x_1 SA1 are $\{D10, D01, D11\}$. Of course, $\{\bar{D}10, \bar{D}01, \bar{D}11\}$ are also propagation D-cubes.

Singular covers are used in test generation to justify the assignments made to the output of an LTG. They are easily obtained from the threshold function of the LTG.

E. Fault Collapsing

To reduce the run-time of test generation, it is necessary to reduce the number of faults in the fault list. This can be done by exploiting fault equivalence and dominance relationships.

Theorem 4: Given an LTG implementing the threshold function, $f(x_1, x_2, \ldots, x_n)$, if there exists an input x_i such that $w_i = T + \delta_{on}$, and if all other inputs x_j have positive weights w_j (i.e., $w_j > 0$), then x_i SA1 is equivalent to f SA1.

Theorem 5: Given an LTG implementing the threshold function, $f(x_1, x_2, \ldots, x_n)$, the following fault dominance relationships hold:

1) An output f SA0 (SA1) dominates an x_i SA0 (SA1) if Equation (3a) is satisfied.
2) An output f SA1 (SA0) dominates an x_i SA0 (SA1) if Equation (3b) is satisfied.

To demonstrate Theorems 4 and 5, consider the threshold function, $f(x_1, x_2, x_3) = x_1x_2 + x_1x_3$ again. Applying the theorems, we see that x_1 SA0 and f SA0 are equivalent. Therefore, either fault can be dropped from the fault list. Also, f SA0 (SA1) dominates x_1 SA0 (SA1), x_2 SA0 (SA1) and x_3 SA0 (SA1). Hence, f SA0 and f SA1 can also be discarded from the fault list.

Exploiting Theorem 5 leads to the following theorems on test generation for irredundant combinational threshold networks. These theorems are similar to those for Boolean testing.

Theorem 6: In an irredundant, fanout-free combinational threshold network, G, any test set, $\mathcal{V}$, that detects all SSFs on the primary inputs detects all SSFs in G.

Theorem 7: In an irredundant combinational threshold network, G, any test set, $\mathcal{V}$, that detects all SSFs on the primary inputs and fanout branches detects all SSFs in G.

V. ATPG Methodology

In this section, we present our test generation framework for combinational threshold networks. The input is a Boolean network and the output is a test set which detects a maximal number of faults. The Boolean network is synthesized into a threshold network using TELS [5]. Then, an equivalent gate-level network is derived by replacing each LTG in the network with its two-level AND-OR equivalent. Next, we generate the fault list using the fault collapsing techniques we have developed and supply it to the SAT-based ATPG

TABLE I
TEST GENERATION ON MCNC BENCHMARKS

Benchmark	No. of SSFs	No. of SSFs with fault collapsing	Reduction (%)	Test set size	Fault coverage (%)	Test efficiency (%)	Time (s)
example2	1,148	876	23.7	73	95.4	100	8
i3	1,192	776	34.8	125	100	100	8
apex6	2,608	2,128	18.4	200	100	100	21
rot	2,700	2,220	17.7	212	97.8	100	18
x3	3,298	2,744	16.8	145	91.9	100	23
frg2	3,316	2,594	21.8	189	95.1	100	27
pair	6,278	5,042	19.7	312	95.3	100	57
i4	1,056	616	41.7	133	100	100	46
i2	1,790	1,016	43.2	207	100	100	11
des	10,872	8,272	23.9	178	99.2	100	142

engine of SIS [6] to do test generation. We then perform threshold logic fault simulation on the test set derived to determine if all the faults for which tests were found were indeed tested. Note that although an equivalent gate-level network is used for threshold network ATPG, the set of stuck-at faults targeted is totally different than what would be targeted in Boolean testing, owing to very different fault collapsing techniques.

VI. EXPERIMENTAL RESULTS

We present experimental results to validate our ATPG methodology for combinational threshold networks in this section. We used the MCNC benchmarks in our experiments, which were conducted on a Dell PowerEdge 600SC server with 768MB RAM running Red Hat Linux 8.0. All the benchmarks were first synthesized into threshold networks using TELS [5]. Due to space restrictions, results for only ten benchmarks are reported.

Tables I shows the effect of using fault equivalence and dominance relationships that we developed in Section IV. It can be seen that on average there is a 26.5% reduction in the number of faults that need to be targeted in the networks. Thus, fault collapsing is a valuable technique that should also be exploited in ATPG for threshold networks. Note that this reduction is not as high as that that can be achieved in Boolean networks (approximately 50%), since very few universal fault equivalence and dominance relationships exist for threshold gates.

In the next set of experiments, we performed test generation using the framework already present in SIS [6]. We modified the test engine and supplied it with our fault list for threshold networks. We then performed threshold fault simulation on the generated vectors to obtain the fault coverage. Tables I shows the number of faults which were targeted during test generation, test set size, fault coverage, test efficiency, and total system execution time. Test efficiency is the ratio of the fault tested or declared redundant and the total number of faults being targeted while fault coverage is the percentage of targeted faults for which test vectors were found. We see that 100% test efficiency is obtained in all cases. Furthermore, the test generation times are very small.

VII. CONCLUSIONS

As the revolution started by CMOS is likely to conclude in the coming decade, emerging nanoscale technologies are being researched for CMOS replacement or integration. Some of these technologies, in particular RTDs and QCA, implement threshold logic, which provides better logic depth and area than Boolean logic. We have introduced an ATPG framework for combinational threshold networks by describing all of its key components of test generation. We showed how faults in an RTD-based gate can be modeled using the stuck-at fault model and how test vectors can be obtained. We related irredundant threshold networks to test generation and developed some theorems on fault collapsing. We implemented our framework in SIS and achieved excellent results. Threshold logic is once again becoming an active area of research (given the promise of future technologies implementing it) and we hope that others in the design automation and testing communities will join in our efforts.

ACKNOWLEDGMENT

The authors gratefully acknowledge the help of Dr. Werner Prost of Gerhard-Mercator-Universität Duisburg for supplying the RTD HSPICE model.

REFERENCES

[1] "Semiconductor Industries Association Roadmap." http://public.itrs.net

[2] K. J. Chen, K. Maezawa, and M. Yamamoto, "InP-based high-performance monostable-bistable transition logic elements (MOBILE's) using integrated multiple-input resonant-tunneling devices," *IEEE Electron Device Lett.*, vol. 17, no. 3, pp. 127–129, Mar. 1996.

[3] K. Maezawa, H. Matsuzaki, M. Yamamoto, and T. Otsuji, "High-speed and low-power operation of a resonant tunneling logic gate MOBILE," *IEEE Electron Device Lett.*, vol. 19, no. 3, pp. 80–82, Mar. 1998.

[4] C. Pacha *et al.*, "Resonant tunneling device logic circuits," University of Dortmund and Gerhard-Mercator University of Duisburg, Tech. Rep., July 1999.

[5] R. Zhang, P. Gupta, L. Zhong, and N. K. Jha, "Synthesis and optimization of threshold logic networks with application to nanotechnologies," in *Proc. Design Automation & Test in Europe Conf.*, Feb. 2004, pp. 904–909.

[6] E. M. Sentovich *et al.*, "Sequential circuit design using synthesis and optimization," in *Proc. Int. Conf. Computer Design*, Oct. 1992, pp. 328–333.

[7] S. Muroga, *Threshold Logic and its Applications*. New York, NY: John Wiley, 1971.

[8] N. K. Jha and S. Gupta, *Testing of Digital Systems*. Cambridge, UK: Cambridge University Press, 2003.

[9] V. Beiu, J. M. Quintana, and M. J. Avedillo, "VLSI implementations of threshold logic - A comprehensive survey," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 1217–1243, Sept. 2003.

[10] W. Prost *et al.*, "Manufacturability and robust design of nanoelectronic logic circuits based on resonant tunneling diodes," *Int. J. Circ. Theory Appl.*, vol. 28, no. 6, pp. 537–552, Nov. 2000.