PATH DELAY FAULT TESTING OF BENES MULTISTAGE INTERCONNECTION NETWORKS

H. T. Vergos[1,2], *M. Bellos*[1] *& D. Nikolos*[1,2]

[1]Dept. of Computer Engineering and Informatics, University of Patras, 26 500, Rio, Greece
[2]Computer Technology Intsitute, 3, Kolokotroni Str., 262 61 Patras, Greece
e-mail : tarka@ceid.upatras.gr, {vergos, nikolosd}@cti.gr

ABSTRACT

In this paper we present two methods for path delay fault testing of circuit-switched Benes Multistage Interconnection Networks (MINs) with centralized control. Although the number of paths is $O(n^3)$, the first method exploiting the inherent parallelism of the Benes MIN requires $O(n^2)$ pairs of test vectors. In the second method we propose the selection of a minimal subset of paths, that are robustly testable by only $O(\log_2 n)$ test vector pairs. The delay along all other paths can be calculated based on the selected path delays.

1. INTRODUCTION

Many kinds of Multistage Interconnection Networks (MINs) have been proposed in the open literature [1]. Blocking MINs provide only one path for connecting each source to a destination. Non-blocking MINs on the other hand offer more that one alternative paths for a source - destination pair to get connected. Benes is a well known non-blocking MIN.

A MIN consists of alternating stages of links and switches. MIN testing has been considered with respect to various fault models [1, 2]. In the state stuck-at fault model, a failure causes a switching element to remain in a particular state. In the link fault model, a failure affects an individual link of a switching element, leaving the remaining part of the switch operational. In the switch fault model a failure makes a switch totally unusable. However, there are physical defects that can degrade the performance of an integrated circuit without altering its logic functionality. Such a degradation is currently modeled by three fault models.

Gross delay faults [3], model delay defects that affect single lines in the circuit, causing the propagation delay through them to be "very large". The gate delay fault model [4] also addresses defects affecting single lines, however, no assumption is made on the delay size. The path delay fault model [5] addresses distributed, or accumulated delays due to propagation through several lines, each affected by a delay effect.

In delay fault test generation we associate two logical paths with each physical path. A logical path is a pair (T, p) with $T=\bar{x} \rightarrow x$, $x \in B=\{0, 1\}$, being a transition at the input of p. In the case of delay fault testing the test set consists of pairs of vectors. Throughout the paper the term test session is used to denote the application of a test vectors pair. Since the number of paths in a contemporary circuit is prohibitively large for all the paths to be tested various path selection methods have been proposed to reduce the paths that must be tested although none of them has proven satisfactory for the general case (see for example [6-8]).

In this paper we address the problem of path delay fault testing of the non-blocking circuit-switched Benes MINs with centralized control. We consider that the network has been implemented as a set of b/M M-bit slices [9], where b is the size of the bus of each source and destination of the network, $1 \leq M \leq b$ and each slice has been implemented as a VLSI chip. For M = b the network has been implemented on a single chip.

2. BENES MINs

We consider nxn Benes MINs, where $n=2^k$. A nxn Benes MIN is constructed from $N=2^*\log_2 n-1$ stages of switches, labeled from left to right from 1 to $2^*\log_2 n-1$. Each stage has (n/2) 2x2 switches. The source and destination nodes are labeled with 0 and $2^*\log_2 n$ respectively. Fig. 1 presents the 8x8 Benes MIN. Distinct paths of the MIN may have common links and switches, but since there are (n/2) paths linking any source to any destination, a conflict appears only when two or more sources are trying to connect to the same destination.

Each switch S, as shown in Fig. 2.*a* has a pair of input data buses X_0, X_1, a pair of output data buses Y_0, Y_1 and a control signal c. All buses are identical in size and unidirectional. The two states of the switch S are determined by the control line c as shown in Fig. 2.*b* and 2.*c*. Each switch is constructed from 2M 2->1 multiplexers, where M is the size of the buses. Each pair of multiplexers (Fig. 2.*d*) accepts two lines of X_0, X_1 buses, the control signal c and drives the corresponding lines of buses Y_0, Y_1.

Without loss of generality, each link is considered as a single virtual line, no matter if it really represents either one physical line or a physical bus. For testing purposes any value applied on a virtual line is applied to every line of the physical bus that it may represent. We will present our analysis considering virtual lines only. The analysis will then be valid for all M lines of the bus. We note that to every virtual path correspond two logical virtual paths.

In an nxn Benes MIN we divide the paths in two sets P and L. P includes all paths starting from a source while L all paths starting from a control input. Since the connections of sources to destinations change dynamically during system operation, delays along paths of L are also significant. Since there are n sources and there

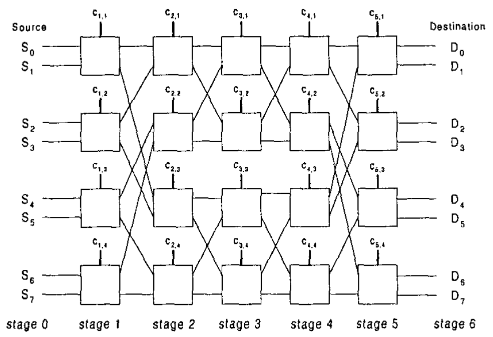

Fig. 1: 8x8 Benes Network

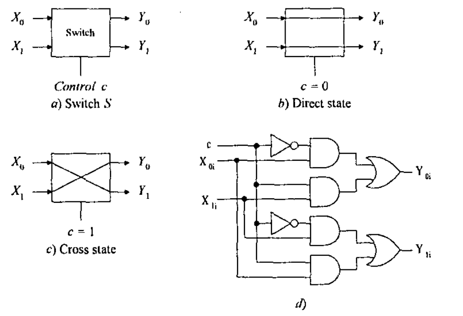

Fig. 2

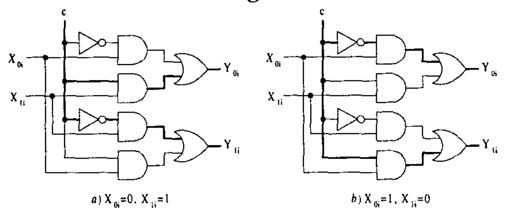

Fig. 3

are (n/2) paths from each one of them to any destination, the cardinality of P, denoted $|P|$, is: $|P| = (n^3/2)$.

Fig. 3.*a* and 3.*b* present the sub-paths from the control input of a switch through two of its 2M multiplexers for X_{0i}=0 and X_{1i}=1 and for X_{0i}=1 and X_{1i}=0. Since the transition, during delay testing, propagates from the control input through 2M multiplexers and along the lines of the bus, in this case we refer also to virtual paths. We observe that at every stage the control input of a switch can be seen as the root of two full binary trees having the destinations as leaves. Each such tree has a depth of N-*i*+1, where *i* is the number of the stage and $i \in \{1, 2, ..., N\}$. Every such tree has 2^{N-i+1} leaves which is also the number of virtual paths. Since each stage has (n/2) switches and each switch is the root of two trees, there are $2^*(n/2)^*2^{N-i+1}$ virtual paths starting from each stage's control inputs. Thus the cardinality of L is equal to the sum of the virtual paths starting from every control input, that is, $|L| = \sum_{i=1}^{N} 2\frac{n}{2}2^{N-i+1} = 2\frac{n}{2}2(2^N-1) = n^3-2n$.

The total number of virtual paths is equal to $|P| + |L| = (3/2)n^3-2n$, the number of the logical virtual paths is $|LV| = 2^*(|P|+|L|) = 3n^3-4n$ and the number of physical lines is $M^*[(3/2)n^3-2n]$. The propagation delays along the M lines of the bus are measured in parallel.

3. METHOD 1: PARALLEL TESTING

For the rest of the paper we consider that all control inputs of switches of stage i, with $i \in \{1, 2, ..., N\}$, of the Benes network take the same value c_i. Since each path in a Benes network is established for a distinct combination of the control inputs, two sets of values of $c_1c_2...c_N$ and $c_1'c_2'...c_N'$, with $c_1c_2...c_N \neq c_1'c_2'...c_N'$, do not have any common path. This means that every control input combination establishes n distinct paths. Then by applying to $c_1c_2...c_N$ all possible values, that is all 2^N different values, the total number of distinct established paths is : $n^*2^N = (n^3/2) = |P|$. Moreover, since only n paths can be established by any $c_1c_2...c_N$ combination for establishing all the paths of set P, all 2^N $c_1c_2...c_N$ combinations are required.

For each $c_1c_2...c_N$ value we can measure the delays along n virtual paths, hence 2^*2^N test sessions are required in order to measure the delays along all logical virtual paths from sources to destinations. If T_p denotes the number of test sessions required to measure the delays along all paths from sources to destinations, $T_p = 2^*2^N = n^2$.

For any $c_{i+1}c_{i+2}...c_N$ value two paths starting from the control input of each switch of the stage i are established. This means that $2^*(n/2)$ = n virtual paths, whose delay can be measured in parallel, are established by any control signal combination. Since to each switch of the stage i correspond two trees each one with 2^{N-i+1} virtual paths we conclude that $2^*2^*2^{Ni+1}/2$ test sessions are required for measuring the propagation delay along the logical virtual paths starting from a control input of stage i. Thus, the total number of test sessions T_L for measuring the propagation delays along the virtual paths starting from control inputs are: $T_L = 2^* \sum_{i=1}^{N} 2^{N-i+1} = 4(2^N-1) = 2n^2-4$. Combining the above we get $T_p + T_L = 3n^2 - 4$.

The above implies that although the number of virtual paths of a nxn Benes network is $O(n^3)$ the number of the required test sessions is $O(n^2)$ considering only the inherent parallelism of the network. In a circuit with n outputs the maximum number of paths that can be robustly tested in parallel is equal to n. Then taking into account that the number of all logical virtual paths of the nxn Benes network is equal to $3n^3$-4n and the fact that $3n^2$–4 test sessions are required we conclude that this is the optimal number of test sessions required to robustly test all possible paths.

4. METHOD 2 : PATH SELECTION BASED METHOD

It has been shown in [6] that by measuring the delays along a suitable very small set Δ of physical paths, the propagation delay along any other path can be calculated. However, this method can not exploit the inherent parallelism of Benes networks. The method proposed in this section exploiting the parallelism of the Benes networks derives a basis with cardinality n times smaller than that derived by the method of [6].

4.1 Set P

We represent the MIN as a graph where each switch, source and destination is represented by a node of the graph and each link by a line. The graph is a collection of full binary trees with root nodes connected by links at stages $m=\log_2 n-1$ and $m+1=\log_2 n$. The binary trees on the left are of depth m, while the ones on the right are of depth m+1.

Fig. 4 presents one pair of the above trees along with their interconnection. We denote a pair of such trees as a t-structure. In every nxn Benes network there are exactly n different t-structures since there are n different links between the stages labeled m and m+1. These t-structures do not have any virtual paths in common because the connection between any two tree pairs that form a t-structure is distinct.

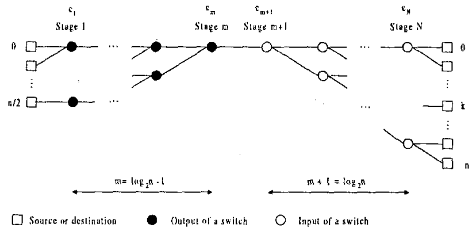

Fig. 4

Due to space limitations, we will not present the proofs of the Lemmas and the Theorems used.

Lemma 1. The n t-structures represent all virtual paths of set P.

Any two virtual paths of a t-structure cannot be tested in parallel since they require at least one of the switches at stages m and m+1 to be in contradictory states. Virtual paths belonging to different t-structures can be tested in parallel for delay faults, provided that two or more virtual paths do not force common switches in contradictory states. At most, n virtual paths one from each t-structure can be tested in parallel by a single test session.

We define Q as the set of the following paths of a t-structure:

a) All paths from one source to all destinations. We denote this set as Q_A.

b) All paths from all sources except the one assumed in a) to a single destination. We denote this set as Q_B.

Theorem 1. If the propagation delays along all paths of set Q of a t-structure are known, the propagation delay along any other path of the t-structure can be calculated.

There is no need though to measure the propagation delays along all virtual paths belonging to Q set for every t-structure. The n t-structures of the network can be split in two parts: the left will contain the left trees and the right the right trees. All left trees have the same depth, as well as all right trees have the same depth. The fact that two t-structures may have common switches that form subtrees on either the right or the left part, as shown in Fig. 5, can be efficiently used for selecting a subset of paths along which the propagation delays must be measured. Let p_1 and p_2 be two sources in two different t-structures TS_1 and TS_2, as shown in Fig. 5. The two t-structures have common switches that form a subtree of *l* levels. Suppose that the delays along all virtual paths starting from p_1 and ending at the destinations in set S_1 and those along all virtual paths starting from p_2 and ending at the destinations of set S_2 are known. Then measuring the delay along a virtual path starting from p_1 and ending at a destination of S_2 and a path starting from p_2 and ending at a destination of S_1 we can calculate the delays along all virtual paths starting from p_1 or p_2 and ending at any destination in sets S_1 and S_2. Suppose that we want to calculate the propagation delay along path p_1->j, where j is a destination in S_2. Knowing the propagation delay along p_1->k, with k in S_2, the propagation delay of p_1->j can be calculated as $d(p_1\text{->}j)=d(p_2\text{->}j)+d(p_1\text{->}k)-d(p_2\text{->}k)$.

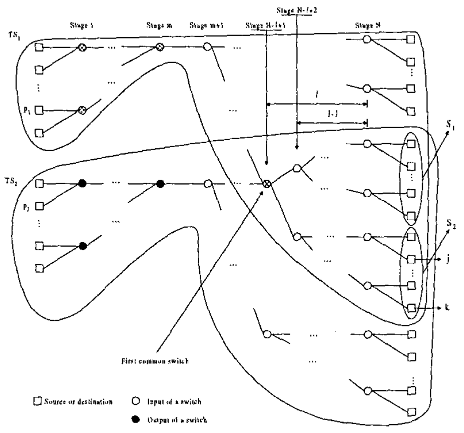

Fig. 5

We will present an algorithm that manipulates the control signal values of the switches, such that n virtual paths, each belonging to a distinct t-structure, can be tested in parallel. The algorithm first manipulates the switches of the right part and next those of the left part. Each stage's control signals are set by a bit of an N-bit binary number. Control signals of stage i, i∈ {1, 2, ..., N} are controlled by the bit c_i. Hence the m leftmost bits represent the control signals' values for the left part of the MIN while the rest m+1 for the right part.

Algorithm 1

```
Step 1. Set $c_1 \ldots c_m c_{m+1} .. c_N = 0 \ldots 00$.
Step 2. Apply two test sessions, one for transition 0->1 and the other for 1->0.
Step 3. Set $c_m c_{m+1} .. c_N = 0 \ldots 01$.
Step 4. Apply two test sessions, one for transition 0->1 and the other for 1->0.
Step 5. Shift left $c_m c_{m+1} .. c_N$ (consider that the rightmost bit is filled with a zero).
Step 6. If $c_m \neq 1$ then go to step 4.
Step 7. Set $c_1 c_2 \ldots c_m c_{m+1} = 100 \ldots 0$
Step 8. Apply two test sessions, one for transition 0->1 and the other for 1->0.
```

MIN	Number of logical virtual paths T	Number of test sessions		Reductions		
		method one T_1	method two T_2	$\frac{T-T_1}{T}100\%$	$\frac{T-T_2}{T}100\%$	$\frac{T_1-T_2}{T_1}100\%$
16x16	12224	764	44	93,75	99,64	94,24
32x32	98176	3068	56	96,88	99,94	98,17
64x64	786176	12284	68	98,44	99,99	99,45
128x128	6290944	49148	80	99,22	100,00	99,84
256x256	50330624	196604	92	99,61	100,00	99,95

Table 1: Comparison results.

Step 9. Shift right $c_1\ c_2\ \dots\ c_m\ c_{m+1}$ (consider that the leftmost bit is filled with a zero).

Step 10. If $c_{m+1} \neq 1$ then go to step 8, else end.

The above algorithm applies $2[m+(m+1)+1] = 4\log_2 n$ test sessions.

Theorem 2. The propagation delays along any path of set P that has not been measured during the application of the algorithm can be calculated from the measured propagation delays.

4.2. Set L

For path delay fault testing of paths starting from the control input of a switch, the inputs X_0 and X_1 of the switch must be set to complementary values. Fig. 3.*a* and 3.*b* present the sub-paths from the control input of a switch through two of its 2M multiplexers for $X_{0i}=0$ and $X_{1i}=1$ and for $X_{0i}=1$ and $X_{1i}=0$ respectively. Therefore for path delay fault testing of the paths starting from the control input of a switch at least two sessions are required, one with $X_{0i}=0$ and $X_{1i}=1$ and one with $X_{0i}=1$ and $X_{1i}=0$.

The following Algorithm establishes virtual paths along which the propagation delays must be measured.

Algorithm 2.

Step 1. Set i=1.

Step 2. Set $c_j=0$ for all $j \in \{1, 2, \dots, N\}$ with $j \neq i$.

Step 3. Set the sources to the suitable values such that each switch to receive $X_0=0$ and $X_1=1$ and measure the delays along the paths from the (n/2) c_i inputs to the n destinations.

Step 4. Set each source to its complement and measure the delays along the paths from the (n/2) c_i inputs to the n destinations.

Step 5. If i<N then set i=i+1 and go to step 2, else end.

For each one of the steps 3 and 4 two measurements are required, one for the transition 0->1 and one for transition 1->0. Therefore Algorithm 2 applies $4*(2\log_2 n - 1) = 8\log_2 n - 4$ test sessions.

After the application of Algorithm 2 we have measured the propagation delays along all paths starting from a control input $c_{i,j}$ of the switch i,j of stage i that pass through the output $Y_0(i,j)$ or $Y_1(i,j)$ of the switch and ends at a destination D_k with $c_{i+1,j_1}, c_{i+2,j_2}, \dots, c_{N,j_k} = 0$, where $j_1, j_2, \dots, j_k \in \{1, 2, \dots, n/2\}$.

Theorem 3. The propagation delays along any virtual path of set L that has not been measured can be calculated from the measured propagation delays, during the application of Algorithm 2, and the propagation delays along paths of set P.

From Theorems 2 and 3 we conclude that applying $12\log_2 n - 4$ test sessions all the information required for calculating the propagation delays along all paths in sets P and L is available.

5. CONCLUSIONS

In this paper, we have shown that an nxn circuit - switched Benes MIN with centralized control has $3n^3 - 4n$ logical paths and presented two methods for delay fault testing of them. The first, which only considers the inherent parallelism of the Benes MIN requires $3n^2 - 4$ test sessions. For the application of the first method we only need to verify that the outputs have the correct value one clock period after the second test vector of each test vector pair is applied. The second method, selects a very small set of paths that require only $12\log_2 n - 4$ test vector pairs for path delay fault testing. The propagation delay along all the remaining paths can be calculated in a straightforward manner by the delays along the selected set. The application of the second method requires the measurement of the propagation delays along the selected paths, therefore the application of this method requires a more aggressive tester. However, for large values of n the number of test vector pairs required by the second method is impressively smaller than that required by the first method; hence the second method is preferable. Table 1 presents comparison results.

6. REFERENCES

[1] H. J. Siegel, *Interconnection Networks for Large - Scale Parallel Processing.* 2nd ed., McGraw-Hill, 1990.

[2] D. P. Agrawal, "Testing and Fault Tolerance of Multistage Interconnection Networks", Computer, pp. 41–53, April 1982.

[3] Z. Brasilai and B. Rosen, "Comparison of ac self-testing procedures", *ITC-83*, pp. 560-571.

[4] J.L. Carter, et.al, "Efficient Test Coverage Determination for Delay Faults", *ITC-87,* pp. 418-427.

[5] G. L. Smith, "Model for delay faults based upon paths", *ITC-85*, pp. 342-349.

[6] J. D. Lesser and J. J. Shedletsky, "An Experimental Delay Test Generator for LSI Logic", IEEE Trans. on Computers, C-29 (3), March 1980, pp. 235 – 248.

[7] S. Tani, et. al., "Efficient Path Selection for Delay Testing Based on Partial Path Evaluation", *16th IEEE VLSI Test Symp.*, pp. 188-193, 1998.

[8] T. Haniotakis, et. al., "C-Testable One - Dimensional ILAs with Respect to Path Delay Faults : Theory and Applications"; *IEEE DFT '98*, pp. 155–163, 1998.

[9] M. A. Franklin, et. al.,, "Pin Limitations and partitioning of VLSI Interconnection Networks", *IEEE Trans. on Computers*, C-31, pp. 1109–1116, November 1982.