Lecture Notes in Computer Science 1667

Jan Hlavička Erik Maehle
András Pataricza (Eds.)

Dependable Computing – EDCC-3

Third European Dependable Computing Conference
Prague, Czech Republic, September 1999
Proceedings

Springer

Path Delay Fault Testing of a Class of Circuit-Switched Multistage Interconnection Networks

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

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

Abstract. In this paper we consider path delay fault testing of a class of isomorphic Multistage Interconnection Networks (MINs) with centralized control using as representative the nxn Omega network. We show that the number of paths is $3n^2-2n$ and we give a method for testing those applying only $2(3n-2)$ pairs of test vectors. We also show that this is the least number of test vector pairs that are required for testing all paths of the MIN. We also give a path selection method such that: a) the number of selected paths, that is, the number of paths that must be tested, is a small percentage of all paths and the propagation delay along every other path can be calculated from the propagation delays along the selected paths, b) all the selected paths are tested by using $2(3\log_2 n+1)$ test vector pairs. Both methods derive strong delay–verification test sets.

1 Introduction

Multistage Interconnection Networks (MINs) represent a compromise between the single bus and the crossbar switch interconnections from the point of view of implementation complexity, cost, connectivity and bandwidth. A MIN consists of alternating stages of links and switches. Many kinds of MINs have been proposed and built for use in massively parallel computers [1, 2].

The testing of MINs has been widely considered with respect to the state stuck-at fault model, the link fault model and the switch fault model [for example 3, 4]. However, physical defects in integrated circuits can degrade circuit performance without altering their logic functionality. Apart from this, increasing performance requirements of the contemporary VLSI circuits makes it difficult to design them with large timing margins. Thus imprecise delay modeling and the statistical variations of the parameters during the manufacturing process may result in circuits with greater delays than the expected ones. The change in the timing behavior of the circuit is modeled by two popular fault models. One is the gate delay fault model where delays violating specifications are assumed to be due to a single gate delay [5, 6]. The other is the path delay fault model where a path is declared faulty if it fails to propagate a transition from the path input to the path output within a specified time interval [7]. The latter model is deemed to be more general since it captures the cumulative effect

J. Hlavicka et al (Eds.): EDCC-3'99, LNCS 1667, pp. 267-282, 1999

of small delay variations in gates along a path as well as the faults caused by a single gate.

A physical path of a circuit is an alternating sequence of gates and lines leading from a primary input to a primary output of the circuit. 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. The cardinality of the test set, that is, the number of pairs of vectors depends on the number of the paths that must be tested and the percentage of the paths that can be tested in parallel. Throughout the paper the term test session is used to denote the application of a test vector pair. The number of physical paths in a contemporary circuit is prohibitively large in order for all the paths to be tested for path delay faults. To this end to reduce the paths that must be tested for path delay faults various path selection methods have been proposed (for example [8–11]) although none of them has proven to be satisfactory for the general case.

In this paper we address the problem of testing for path delay faults a class of isomorphic circuit-switched MINs with centralized control, using as representative the Omega network [12]. We consider that the network has been implemented as a set of b/M M-bit slices [13], 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 (probably on a single wafer). The test sets that we derive are strong delay–verification test sets, therefore, their application ensures that if the circuit under test (CUT) functions correctly at a speed it will also operate correctly at every lower speed. In section 2 we present the terminology that will be used in this paper. In section 3 we present the main features of the Omega network and we show that the number of physical paths is $O(n^2)$. In section 4 exploiting the inherent parallelism of the Omega network we show that it can be tested for path delay faults in O(n) test sessions. We also show that the derived test set is a strong delay-verification test set. In section 5 we present a new path selection method such that: a) the paths, which are selected for testing constitute a small percentage of the total number of paths; the delay along the rest paths can be calculated from the propagation delays along the selected paths, b) the propagation delays along n of the selected paths are measured in parallel during each test session. According to this method the required number of test sessions is $O(\log_2 n)$. The application of this method cuts down the test effort as well as the test application time significantly. We also show that the derived test set is a strong delay–verification test set. The conclusions are given in section 6.

2 Preliminaries

A two pattern test $T = \langle V_1, V_2 \rangle$ is said to be a robust delay test for a path P, for a rising or falling transition at the output of the path, if and only if, when P is faulty and test T is applied, the circuit output is different from the expected state at sampling time, independent of the delays along gate inputs not on P [14]. A robust test may

actually propagate transitions to an output through more than one path to that output in the circuit; such a test is called Multiple-Path Propagating Robust Test (MPP-RT) [15]. A robust test that propagates the fault effect through only a single path to an output in the circuit will be called a Single-Path Propagating Robust Test (SPP-RT) for that output. For example consider the circuit in Figure 1 [15]. The test < V_1, V_2>, with V_1=(a=1, b=0, c=1, d=1) and V_2=(a=0, b=0, c=0, d=1) for a falling transition at the output y is a MPP-RT, which sensitizes and propagates fault effects robustly along both the paths a-3-y and c-3-y to the output y. The test <V_1, V_2> with V_1= (a=0, b=0, c=1, d=1) and V_2 = (a=0, b=0, c=0, d=1) for a falling transition at the output y is a SPP-RT, which sensitizes and propagates the fault effect robustly only along the single path c-3-y. We define a robust test as Multiple SPP-RT (M-SPP-RT) if it propagates the effect of one or more faults along distinct paths or along paths starting from the same primary input and ending at distinct outputs without internal reconvergent fanouts. For example in Figure 4.a the test < V_1, V_2>, with V_1= (X_{0i}=0, X_{1i}=0, c=0) and V_2= (X_{0i}=1, X_{1i}=1, c=0), is a M-SPP-RT that propagates the effect of delay faults along the distinct paths X_{0i}-3-7-Y_{0i} and X_{1i}-5-8-Y_{1i}. Also the test <V_1, V_2> withV_1= (X_{0i}=0, X_{1i}=1, c=0) and V_2= (X_{0i}=0, X_{1i}=1, c=1)), is a M-SPP-RT that propagates the effect of delay faults along the paths c-4-7- Y_{0i} and c-2-5-8-Y_{1i}. We have to note that in the circuit of Figure 1 the test < V_1, V_2>, with V_1=(a=0, b=0, c=0, d=1) and V_2=(a=1, b=0, c=0, d=1) is not a M-SPP-RT because the paths a-2-5-x and a-3-5-x reconverge.

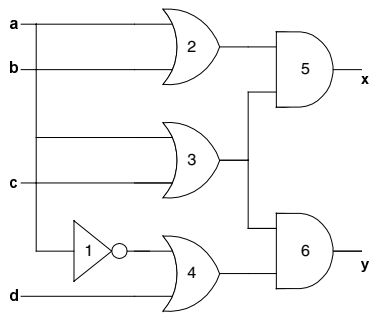

Fig. 1.

A robust test is said to be a Hazard-Free Robust Test (HFRT) if no hazards can occur on the tested path during the application of the test, regardless of the gate delay values. Therefore, a M-SPP-HFRT < V_1, V_2> has to provide steady, glitchless, sensitizing values at all the off-path inputs along more than one paths, when the primary inputs changed from V_1, to V_2.

Robust tests may not exist for all path delay faults in an arbitrary circuit. Some nonrobust tests can be shown to be valid if certain other faults have been tested robustly [16]. Such tests are called Validatable Nonrobust (VNR) tests. The term RV tests is used to denote tests that are robust or validatable nonrobust. A circuit is RV - testable if there is a robust or a VNR test for any single path delay fault. It has been

shown in [17] that the fact that a circuit functions correctly at a clock speed does not imply that it will also function correctly at a lower clock speed. A set of path delay tests is called a strong delay–verification test set if the correct response of the CUT at a speed implies correct operation at any lower speed [17]. A circuit which has a strong delay-verification test set is called a delay-verifiable circuit [17].

3 Omega Networks

We consider nxn Omega MINs, where $n=2^k$. An Omega MIN is constructed from $N=\log_2 n$ stages of switches, where each of the stages has n/2 2x2 switches. The switch stages are labeled from 1 to N. There are also the stages 0 and N+1 which are formed from the source and destination nodes respectively. The interconnection pattern between adjacent stages is the perfect shuffle permutation [18]. This holds for all pairs of stages except N and N+1. Figure 2 shows an 8x8 Omega MIN.

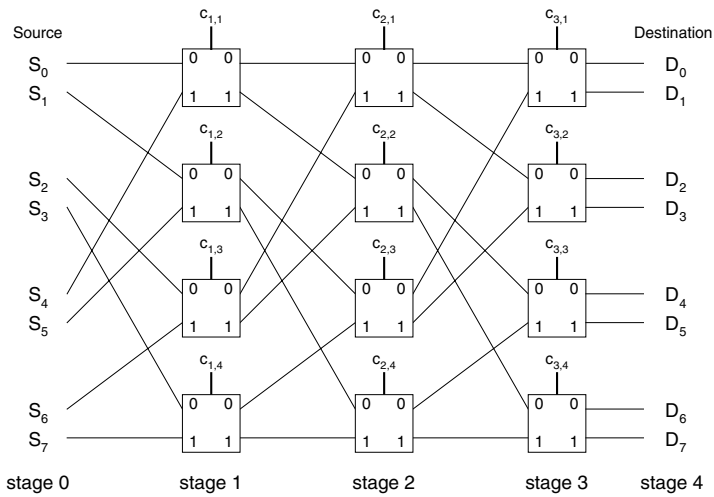

Fig. 2. 8x8 Omega Network

An inherent property of the Omega MIN is that distinct paths of the MIN may have common links and switches. Thus, a conflict appears when any two sources are trying to set any switch of the network in complementary states.

Each switch *S*, as shown in Figure 3.*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 four buses are identical in size and unidirectional. The two states of the switch *S* are determined by the control line *c* as follows: the *direct* state shown in Figure 3.*b*, where the values of X_0, X_1 are propagated to Y_0, Y_1 respectively, and a *cross* state shown in Figure 3.*c*, where the values of X_0, X_1 are propagated to Y_1, Y_0 respectively. The upper input and output are labeled with 0 while the lower are labeled with 1. Each switch is constructed from 2M 2->1 multiplexers, where M is the size of the buses. Each pair of multiplexers, see Figure

3.*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.

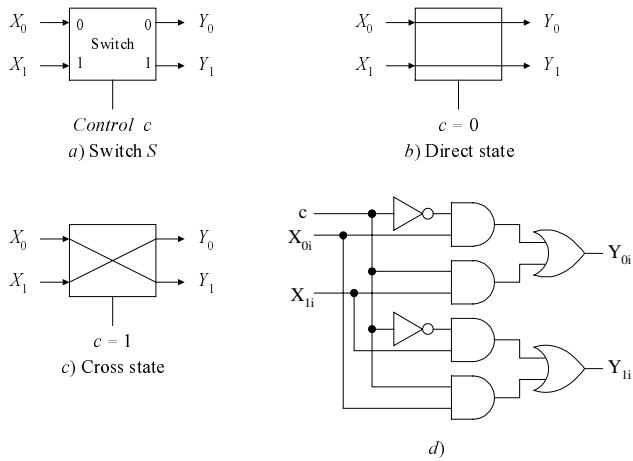

Fig. 3.

Without loss of generality we consider that each link between sources and switches, between switches, as well as switches and destinations is a single virtual line, that may actually represent either one physical line or a physical bus. For testing purposes any value of the virtual line is always applied to every line of the physical bus that it may represent. Paths along the MIN are formed by concatenation of subpaths along links and subpaths through switches of the MIN as well as by subpaths sourcing from the control signals. We will hereafter present the analysis based on virtual lines (or simply lines). The analysis will be valid for all M lines of the bus. We note that to every virtual path or line correspond two logical virtual paths.

In an nxn Omega MIN we distinguish the paths in two sets: those not including subpaths sourcing from a control input and those which do. Since the connections of sources to destinations change dynamically during system operation, delays that stem from the control signals are also significant. Let P be the set consisting of all virtual paths starting from any source and ending at any destination. Since there are n sources and there is only one path from one of them to all the destinations, the number of all possible virtual paths is n^2. That is the cardinality of P, denoted $|P|$, is $|P| = n^2$.

Let L be the set consisting of all paths starting from the control input of a switch and ending at any destination. Figures 4.*a* and 4.*b* present the subpaths 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. For computing the number of the virtual paths starting from

a control input 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 $i \in \{1, 2, ..., N\}$. The latter means that 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, which is: $|L| = \sum_{i=1}^{N} 2\frac{n}{2}2^{N-i+1} = 2\frac{n}{2}2(2^N-1) = 2n(n-1)$

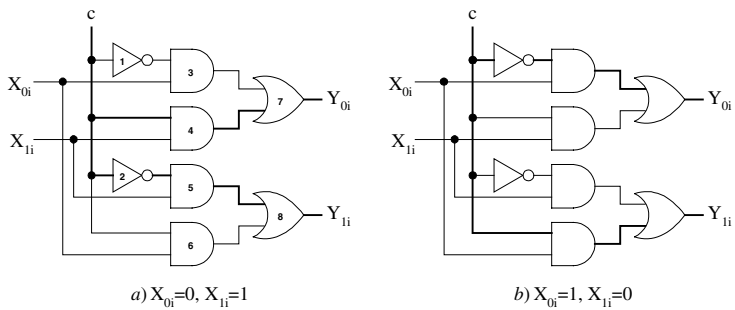

Fig. 4.

Therefore the total number of virtual paths is equal to: $|P| + |L| = 3n^2 - 2n$ and the number of all logical virtual paths is: $|LV| = 2(3n^2 - 2n)$

The number of physical lines is equal to $M(3n^2 - 2n)$, however the propagation delays along the M lines of the bus are measured in parallel.

Although the number of the virtual paths of an nxn Omega network is $O(n^2)$ we will show in the next section that due to the inherent parallelism of the network the propagation delay along various paths can be measured in parallel.

4 Method One: Parallel Testing

It is well known that in an Omega network when the control inputs get a set of values each source is connected to a different destination. Throughout this section we consider that all control inputs of stage i, for i=1, 2, ..., N, of the Omega network take the same value c_i. Then changing the value of the control inputs of a stage i the switches of this stage from the direct state go to the cross state or from the cross state go to the direct state, and all paths from sources to destinations are changed. Then taking into account that in the Omega network only one path exists from a specific source to a specific destination we conclude that for the two sets of values of $c_1c_2...c_N$ and of $c_1'c_2'...c_N'$, with $c_1c_2...c_N \neq c_1'c_2'...c_N'$, a common path from the source to destination in both configurations of the network does not exist. Therefore, applying

to $c_1c_2...c_N$ all possible values, that is 2^N different values, we ensure that each source has been connected to each destination. A feature of the Omega network is that every source can be connected to every destination. Thus taking into account that we have n destinations we conclude that at least n value combinations of the control signals are necessary so that each source to be connected to each destination. From the above discussion we conclude that 2^N is the least number of configurations that ensures that all possible paths from sources to destinations have been established. For each configuration, that is a value of $c_1c_2...c_N$, 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. Let T_p denote the number of test sessions required to measure the delays along all paths from sources to destinations. Then :

$$T_p = 2*2^N = 2n \quad (\mathbf{1})$$

As we have already observed at every stage i a control input can be seen as a root of two full binary trees having the destinations as leaves. We have also shown that every such tree has 2^{N-i+1} virtual paths from the root to the leaves. For any combination of values of $c_{i+1}c_{i+2}...c_N$ two paths starting from the control input of each switch of the stage i are established, that is we have 2*n/2 virtual paths, along which the delay can be measured in parallel. Then taking into account that to each switch of the stage i correspond two trees each one with 2^{N-i+1} virtual paths we conclude that $2 * 2 * 2^{N-i+1} / 2$ test sessions are required for measuring the propagation delay along the logical virtual paths starting from a control input of stage i. Taking into account that i=1, 2, ..., N we get that the total number of test sessions T_L for measuring the propagation delays along the virtual paths starting from control inputs is :

$$T_L = 2 * \sum_{i=1}^{N} 2^{N-i+1} = 4(2^N-1) = 4(n-1) \quad (\mathbf{2})$$

From relations (1) and (2) we get: $T_p + T_L = 2(3n-2)$.

Therefore, while the number of virtual paths of a nxn Omega network is $O(n^2)$ we have shown that the number of the required test sessions is O(n). In a circuit with n outputs the maximum number of paths that can be tested in parallel is equal to n. Then taking into account that the number of all logical virtual paths of the nxn Omega network is equal to $2(3n^2-2n)$ and the fact that 2(3n-2) test sessions are required we conclude that this is the optimal number of test sessions required to test all possible paths.

From the above discussion it is evident that all paths of an Omega MIN are sensitizable. Furthermore it is evident that the derived test set consists of M-SPP-HFRT test vector pairs, that test all paths of the MIN. Since the M-SPP-HFRT tests are a subset of the RV-tests and we test all paths of the MIN, from Theorem 1 in [17] we conclude that the proposed test set is a strong delay-verification test set. This implies that the correct function of the circuit at the tested speed ensures correct operation at any lower speed.

5 Method Two : Path Selection Based Method

It has been shown in [8] that by measuring the delays along a suitable very small set • of physical paths, the propagation delay along any other path can be calculated. However, this method can not exploit the inherent parallelism of Omega or their isomorphic networks. The method proposed in this section exploiting the parallelism of the Omega networks derives a basis with cardinality n times smaller than that derived by the method of [8]. For simplifying the analysis, we examine the sets P, L separately.

5.1 Set P

As a first step, 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. We observe that the graph is a collection of full binary trees with root nodes connected by links at stages m and m+1 with m∈ {1, 2, ..., N-1}. When N is even we choose m=N/2 and the binary trees on the left as well as on the right are of depth m, while when N is odd we choose m = $\lfloor N/2 \rfloor$, where $\lfloor x \rfloor$ denotes the integer part of x, and the binary trees on the left are of depth m, while the ones on the right are of depth m+1. The left trees have as leaves the sources and the right the destinations. We define m' to be the depth of the right tree and it is equal either to m if N even, or to m+1 if N is odd.

Figure 5 presents one pair of the above trees along with their interconnection. We denote a pair of such trees as a t-structure. All t-structures are similar. In every nxn Omega network there are exactly n different t-structures since there are n different links between the stages where the connection of a pair of trees takes place. 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. The latter property does not eliminate the possibility of having common subpaths.

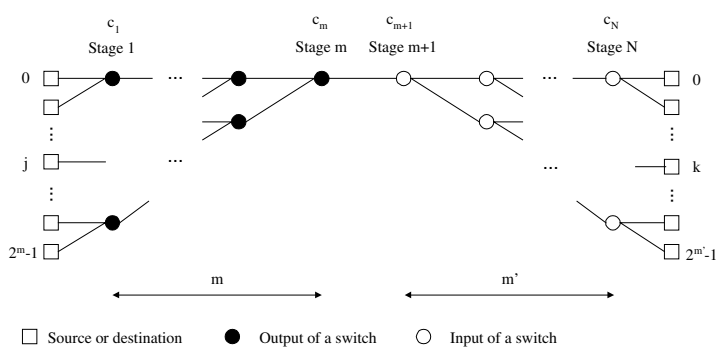

Fig. 5.

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

Proof. a) N is even, that is N=2m. Then the two trees of a t-structure have the same number m of levels and therefore have 2^m leaves each. The number of virtual paths, denoted V, of each t-structure is: $V = 2^m 2^m = 2^{2m} = 2^N = n$ ($N=\log_2 n$)
b) N is odd, that is N=2m+1. Then the left tree of a t-structure has m levels and the right tree has m+1. Thus, they have 2^m and 2^{m+1} leaves respectively. The number of virtual paths of a t-structure is: $V = 2^m 2^{m+1} = 2^{2m+1} = 2^N = n$ ($N=\log_2 n$)
In both cases we have n t-structures and any two of them represent distinct virtual paths, therefore, the number of virtual paths in each case is n^2. ■

Any two virtual paths of a t-structure cannot be tested in parallel for delay faults since every possible pair of virtual paths requires at least one of the switches at stages m and m+1 to be in contradictory states. For example, in Figure 2, the paths from S_0 to D_0 and S_4 to D_2 belong to the same t-structure and the propagation delays along them cannot be measured in parallel because the switches at the stages m=1 and m+1=2 cannot be in the direct and cross state simultaneously. On the contrary, virtual paths belonging to different t-structures can be tested in parallel for path delay faults, provided that two or more virtual paths do not force common switches in contradictory states. If no conflict arises, 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:

- All paths from one source to all destinations. The number of these paths is equal to the number of leaves of the right tree. We denote this set as Q_A.
- All paths from all sources except the one assumed in a) to a single destination. The number of these paths is equal to the number of leaves of the left tree minus 1. 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 path of a t-structure can be calculated.
Proof. Without loss of generality suppose that Q contains all paths from the source with address 0 to all $2^{m'}$ destinations and the paths from sources with addresses 1 to 2^m-1 to the destination with address p, where $0 \le p \le 2^{m'}-1$. Let j->k be a path from source with address j, $0<j\le 2^m-1$ to the destination with address k with $0 \le k \le 2^{m'}-1$ and $k \ne p$. Then the propagation delay along the path j->k can be calculated as :

$$d(j->k) = d(j->p) + d(0->k) - d(0->p)$$ ■

We will show in the sequel that there is no need to measure the propagation delays along all virtual paths belonging to the set Q 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. 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 takes advantage of the fact that two t-structures can have common switches that form subtrees either on the right or the left part, as shown in Figure 6. This can be used in minimizing both the number of virtual paths that need to be tested and the number of test sessions required. 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, $i \in \{1, 2, ..., N\}$ are controlled by bit c_i. Hence the m leftmost bits represent the control signals' values for the left part of the MIN while the rest for the right part.

Let p_1 and p_2 be two sources in two different t-structures TS_1 and TS_2, as shown in Figure 6. 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. For example, suppose that we want to calculate the propagation delay along path p_1->j, where j is a destination in S_2. If we know the propagation delay along p_1->k, k is in S_2, then the propagation delay of p_1->j can be calculated from the propagation delays of p_1->k, p_2->j and p_2->k as : d(p_1 ->j) = d(p_2->j) + d(p_1->k) - d(p_2->k).

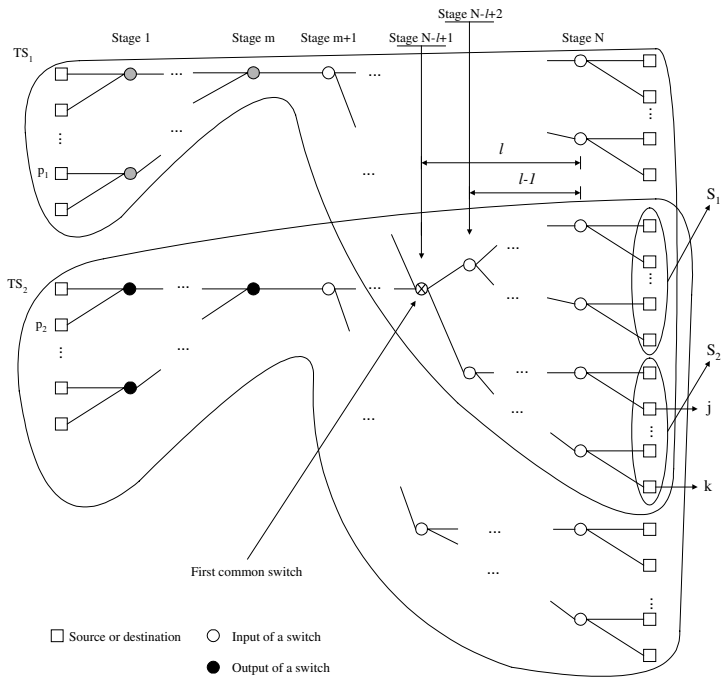

Fig. 6.

The following algorithm establishes all virtual paths along which the propagation delay must be measured.

Algorithm 1

Consider that all control inputs of stage i, for i = 1, 2, ..., N take the same value of c_i. Then one control bit is required for describing the state of the switches of every stage. Hence for the N stages we need N bits, c_1, c_2, ..., c_N. Set m = $\lfloor N/2 \rfloor$ and m' = $\lceil N/2 \rceil$, where $\lceil x \rceil$ denotes the least integer greater than or equal to x.

`Step 1.` Set c_1 ... c_m c_{m+1} .. c_N = 0 ... 00.

Step 2. Apply two test sessions, one for transition 0->1 and the other for 1->0.
Step 3. Set $c_m\ c_{m+1}\ ..\ c_N = 0\ ...\ 01$.
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 $c_1\ c_2 ...\ c_m\ c_{m+1} = 100\ ...\ 0$
Step 8. Apply two test sessions, one for transition 0->1 and the other for 1->0.
Step 9. Shift right $c_1\ c_2\ ...\ c_m\ c_{m+1}$ (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.

From the above algorithm we conclude that $2(m' + m + 1) = 2N + 2$ test sessions are required.

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.

Proof. By steps 1, 2, 3, 4 the propagation delays along 2n virtual paths for n trees of the form of Figure 7.a have been measured. Thus for each tree we know the propagation delays along virtual paths from the same source to two destinations, since switches on the left side of the MIN remain unchanged. These n trees are of depth 1. After the first iteration of steps 5, 6, 4 switches at stage N-1 are considered and the propagation delays along n more virtual paths have been measured. These n virtual paths are distinct compared to the previous virtual paths established since the proposed algorithm manipulates a different stage of switches at each iteration. Thus any of the n virtual paths established from a source end at a distinct destination.

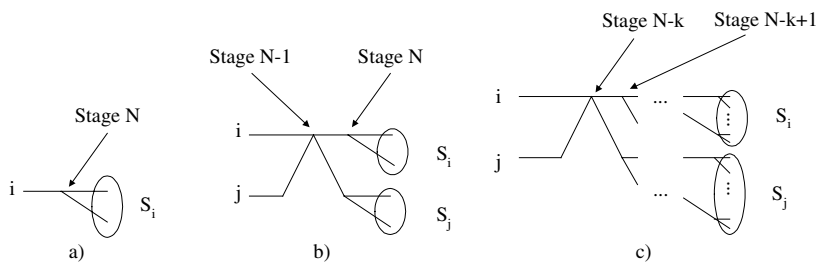

Fig. 7.

Suppose that from source i the virtual path ends at a destination of S_j. Then the virtual path starting from j ends at a destination of S_i since the switch that is manipulated is common for both virtual paths. From the above, we can calculate the propagation delays along all virtual paths starting from i or j and ending at a destination of $S_i \cup S_j$. The latter means that two trees of depth 1, with their right subtrees connected by a switch at stage N-1, can be combined to form two trees of depth 2 from the same sources, as shown in Figure 7.b. This is possible for every pair of such trees and thus after each iteration we have n trees available.

After k-1 iterations of steps 5, 6, 4 the propagation delays along the virtual paths of n trees of depth k have been measured or calculated. Suppose that we consider two

such trees i, j that have a common switch at stage N-k. At the next iteration of steps 5, 6, 4 this switch is set to the cross state. Now a virtual path is established from i to a destination of S_j and another one from j to a destination of S_i. In the same manner we can calculate the propagation delays along all virtual paths from i or j to $S_i \cup S_j$ thus forming two trees of depth k+1 that start from i and j, as shown in Figure 7.c.

After m' iterations the propagation delays along all virtual paths of n m'-depth trees are known which means that we know all the propagation delays along all virtual paths of each of the n Q_A sets.

In the same manner, steps 7-10 of Algorithm 1 provide the information for calculating the propagation delays along all virtual paths of n m-depth trees that each ends at a specific destination. Hence we know the propagation delays along all virtual paths in the Q_B sets. Thus we can calculate the propagation delays along all virtual paths of all the Q_A and Q_B sets and from Theorem 1 we can calculate the propagation delays along all virtual paths of set P. ■

5.2 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.

Figures 4.*a* and 4.*b* present the subpaths 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.

Consider for the rest of this section that all control inputs of stage i, for i=1, 2, ..., N of the network take the same value c_i. 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∈ {1, 2, ..., N} with j≠ 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.

We note that in 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 4N = $4\log_2 n$ 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 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 $c_{i+1,j_1} = c_{i+2,j_2} = ... = c_{N,j_{N-i-1}} = 0$, where $j_1, j_2, \ldots, j_{N-i-1} \in \{1, 2, ..., n/2\}$.

For example, we can see from Figure 2 that such a path is the path from $c_{1,1}$->D_0, which is established when we have $S_0 = 0$ and $S_4 = 1$ (for $S_0 = 1$ and $S_4 = 0$ we have

another path from $c_{1,1}$->D_0) and $c_{2,1} = c_{3,1} = 0$. The propagation delays along the path $c_{1,1}$->D_3, which is established when we have $S_0 = 0$, $S_4 = 1$ and $c_{2,1} = c_{3,1} = 1$, has not been measured. However, the propagation delay along $c_{1,1}$->D_3 can be calculated from the propagation delays along paths $c_{1,1}$->D_0 (was measured), S_0->D_0 and S_0->D_3 as : $d(c_{1,1}\text{->}D_3) = d(S_0\text{->}D_3) + d(c_{1,1}\text{->}D_0) - d(S_0\text{->}D_0)$

We note that the paths S_0->D_0 and S_0->D_3 belong to P, hence the propagation delays along them are already known.

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

Proof. Let $c_{i,j}$ denote the control input of switch j of stage i, then $j \in \{1, 2, ..., n/2\}$. Consider a path that starts from a control input $c_{i,j}$, passes through Y_z, with z = 0 or z = 1 that ends at D_w such that at least one of c_{i+1,j_1}, c_{i+2,j_2}, ..., $c_{N,j_{N-i-1}}$ is equal to 1. The propagation delay along the path $c_{i,j}$ -> Y_z -> D_w has not been measured. Consider the following paths:

The path $c_{i,j}$ -> Y_z -> $D_{w'}$, with $w' \neq w$, such that $c_{i+1,j_1} = c_{i+2,j_2} = ... = c_{N,j_{N-i-1}} = 0$. The propagation delay along this path has been measured during the application of Algorithm 2.

The paths S_x->Y_z->D_w and S_x->Y_z-> $D_{w'}$ (these belong to P, hence the propagation delays along them are already known). Then :

$$d(c_{i,j} \text{->} Y_z \text{->} D_w) = d(S_x\text{->}Y_z\text{->}D_w) + d(c_{i,j} \text{->} Y_z \text{->} D_{w'}) - d(S_x\text{->}Y_z\text{->} D_{w'}) \quad \blacksquare$$

From Theorems 2 and 3 we conclude that with $2(3\log_2 n + 1)$ test sessions we have obtained all the needed information in order to calculate the propagation delays along all paths in sets P and L.

The first method can also be used to derive the maximum speed of the CUT and ensures that the circuit will function correctly for lower speeds. We have shown that the second method can alternatively be used to derive the maximum speed of the CUT. Since the application of the first method ensures that the circuit functions correctly for lower speeds and the maximum speed can alternatively be derived following the second method, we conclude that, when either the first or the second method is used, it is ensured that the CUT will function correctly for all speeds lower than the maximum. Therefore, the test set derived following the second method is also a strong delay-verification test set.

6 Conclusions

We have presented two methods for path delay fault testing of the nxn circuit switched Omega MIN with centralized control. Following the first and the second method respectively $2(3n - 2)$ and $2(3\log_2 n + 1)$ test sessions are required, while the total number of logical paths is equal to $2(3n^2 - 2n)$. The application of the first method requires only verification that the outputs have the correct value one clock period after the application of the second test vector of each test vector pair. 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 significantly smaller than that required by the first method hence the second method is preferable. We present comparison results in Table 1.

Both methods give strong delay-verification test sets, therefore their application ensures that if the CUT functions correctly at a speed it will operate also correctly at all lower speeds.

Table 1. Comparison results.

MIN	Number of logical virtual paths T	Number of test sessions method one T_1	Number of test sessions method two T_2	Reduction $\frac{T - T_1}{T}100\%$	Reduction $\frac{T - T_2}{T}100\%$	Reduction $\frac{T_1 - T_2}{T_1}100\%$
16x16	1472	92	26	93.75	98.23	71.74
32x32	6016	188	32	96.88	99.45	82.98
64x64	24320	380	38	98.44	99.85	90.00
128x128	97792	764	44	99.19	≈100	94.24
256x256	392192	1532	50	99.6	≈100	96.74
512x512	1570816	3068	56	99.8	≈100	98.17
1024x1024	6287360	6140	62	99.9	≈100	98.99

Table 2. Test vectors and established paths for 8x8 Omega network.

	$S_0S_1S_2S_3S_4S_5S_6S_7$	$c_1c_2c_3$	Virtual paths
P	T T T T T T T T*	0 0 0	S_0-D_0, S_1-D_1, S_2-D_2, S_3-D_3, S_4-D_4, S_5-D_5, S_6-D_6, S_7-D_7
	T T T T T T T T	0 0 1	S_0-D_1, S_1-D_0, S_2-D_3, S_3-D_2, S_4-D_5, S_5-D_4, S_6-D_7, S_7-D_6
	T T T T T T T T	0 1 0	S_0-D_2, S_1-D_3, S_2-D_0, S_3-D_1, S_4-D_6, S_5-D_7, S_6-D_4, S_7-D_5
	T T T T T T T T	1 0 0	S_0-D_4, S_1-D_5, S_2-D_6, S_3-D_7, S_4-D_0, S_5-D_1, S_6-D_2, S_7-D_3
L	0 1 0 1 0 1 0 1	0 0 T	$c_{3,1}$-D_0, $c_{3,1}$-D_1, $c_{3,2}$-D_2, $c_{3,2}$-D_3, $c_{3,3}$-D_4, $c_{3,3}$-D_5, $c_{3,4}$-D_6, $c_{3,4}$-D_7
	1 0 1 0 1 0 1 0	0 0 T	$c_{3,1}$-D_0, $c_{3,1}$-D_1, $c_{3,2}$-D_2, $c_{3,2}$-D_3, $c_{3,3}$-D_4, $c_{3,3}$-D_5, $c_{3,4}$-D_6, $c_{3,4}$-D_7
	0 0 1 1 0 0 1 1	0 T 0	$c_{2,1}$-D_0, $c_{2,1}$-D_2, $c_{2,2}$-D_4, $c_{2,2}$-D_6, $c_{2,3}$-D_1, $c_{2,3}$-D_3, $c_{2,4}$-D_5, $c_{2,4}$-D_7
	1 1 0 0 1 1 0 0	0 T 0	$c_{2,1}$-D_0, $c_{2,1}$-D_2, $c_{2,2}$-D_4, $c_{2,2}$-D_6, $c_{2,3}$-D_1, $c_{2,3}$-D_3, $c_{2,4}$-D_5, $c_{2,4}$-D_7
	0 0 0 0 1 1 1 1	T 0 0	$c_{1,1}$-D_0, $c_{1,1}$-D_4, $c_{1,2}$-D_1, $c_{1,2}$-D_5, $c_{1,3}$-D_2, $c_{1,3}$-D_6, $c_{1,4}$-D_3, $c_{1,4}$-D_7
	1 1 1 1 0 0 0 0	T 0 0	$c_{1,1}$-D_0, $c_{1,1}$-D_4, $c_{1,2}$-D_1, $c_{1,2}$-D_5, $c_{1,3}$-D_2, $c_{1,3}$-D_6, $c_{1,4}$-D_3, $c_{1,4}$-D_7

*T denotes a 0->1 and a 1->0 transition.

Although the analysis has been made using the nxn circuit-switched Omega network with centralized control, it is valid for all isomorphic to the Omega networks [12], that can be obtained by suitably permuting switching elements and associated links of the Omega network. As an example in Tables 2 and 3 we give the test vector

pairs, derived from Method two for the 8x8 Omega and Generalized Cube (Figure 8) networks respectively. The virtual paths that are tested in any case can be different, hence the paths along which the propagation delays must be calculated from the measured delays are also different.

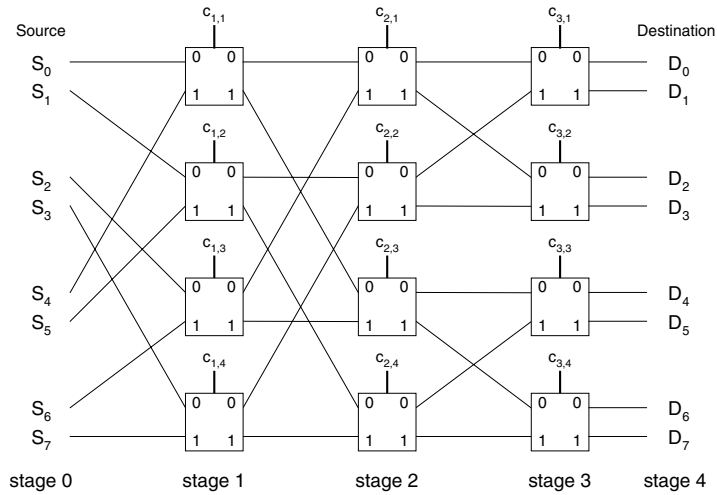

Fig. 8. 8x8 Generalized Cube Network

Table 3. Test vectors and established paths for 8x8 Generalized Cube network.

	$S_0S_1S_2S_3S_4S_5S_6S_7$	$c_1c_2c_3$	Virtual paths
P	T T T T T T T T	0 0 0	S_0-D_0, S_1-D_1, S_2-D_2, S_3-D_3, S_4-D_4, S_5-D_5, S_6-D_6, S_7-D_7
	T T T T T T T T	0 0 1	S_0-D_1, S_1-D_0, S_2-D_3, S_3-D_2, S_4-D_5, S_5-D_4, S_6-D_7, S_7-D_6
	T T T T T T T T	0 1 0	S_0-D_2, S_1-D_3, S_2-D_0, S_3-D_1, S_4-D_6, S_5-D_7, S_6-D_4, S_7-D_5
	T T T T T T T T	1 0 0	S_0-D_4, S_1-D_5, S_2-D_6, S_3-D_7, S_4-D_0, S_5-D_1, S_6-D_2, S_7-D_3
L	0 1 0 1 0 1 0 1	0 0 T	$c_{3,1}$-D_0, $c_{3,1}$-D_1, $c_{3,2}$-D_2, $c_{3,2}$-D_3, $c_{3,3}$-D_4, $c_{3,3}$-D_5, $c_{3,4}$-D_6, $c_{3,4}$-D_7
	1 0 1 0 1 0 1 0	0 0 T	$c_{3,1}$-D_0, $c_{3,1}$-D_1, $c_{3,2}$-D_2, $c_{3,2}$-D_3, $c_{3,3}$-D_4, $c_{3,3}$-D_5, $c_{3,4}$-D_6, $c_{3,4}$-D_7
	0 0 1 1 0 0 1 1	0 T 0	$c_{2,1}$-D_0, $c_{2,1}$-D_2, $c_{2,2}$-D_1, $c_{2,2}$-D_3, $c_{2,3}$-D_4, $c_{2,3}$-D_6, $c_{2,4}$-D_5, $c_{2,4}$-D_7
	1 1 0 0 1 1 0 0	0 T 0	$c_{2,1}$-D_0, $c_{2,1}$-D_2, $c_{2,2}$-D_1, $c_{2,2}$-D_3, $c_{2,3}$-D_4, $c_{2,3}$-D_6, $c_{2,4}$-D_5, $c_{2,4}$-D_7
	0 0 0 0 1 1 1 1	T 0 0	$c_{1,1}$-D_0, $c_{1,1}$-D_4, $c_{1,2}$-D_1, $c_{1,2}$-D_5, $c_{1,3}$-D_2, $c_{1,3}$-D_6, $c_{1,4}$-D_3, $c_{1,4}$-D_7
	1 1 1 1 0 0 0 0	T 0 0	$c_{1,1}$-D_0, $c_{1,1}$-D_4, $c_{1,2}$-D_1, $c_{1,2}$-D_5, $c_{1,3}$-D_2, $c_{1,3}$-D_6, $c_{1,4}$-D_3, $c_{1,4}$-D_7

The application of the proposed methods to a wider class of MINs (for example Delta and Banyan MINs) as well as their extension for path delay fault diagnosis are under investigation.

References

1. H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing*, 2nd ed., New York: McGraw-Hill, 1990.
2. T. Feng, "A survey of Interconnection Networks", Computer, pp. 12–27, December 1981.
3. D. P. Agrawal, "Testing and Fault Tolerance of Multistage Interconnection Networks", Computer, pp. 41 – 53, April 1982.
4. V. P. Kumar and S. M. Reddy, "Augmented shuffle-exchange multistage interconnection networks", Computer, pp. 30 – 40, June 1987.
5. Z. Brasilai and B. Rosen, "Comparison of ac self-testing procedures", Proc. of ITC-83, pp. 560-571.
6. K. D. Wagner, "The error latency of delay faults in combinational and sequential circuits", Proc. of ITC-85, pp. 334 - 341.
7. G. L. Smith, "Model for delay faults based upon paths", Proc. of ITC-85, pp. 342 - 349.
8. J. D. Lesser and J. J. Shedletsky, "An Experimental Delay Test Generator for LSI Logic", IEEE Trans. on Computers, vol. C-29 (3), pp. 235 – 248, March 1980.
9. W. K. Lam, et al., "Delay fault coverage, test set size and performance trade-offs", IEEE Trans. on CAD, vol. 14 (1), pp. 32 - 44, Jan. 1995.
10. S. Tani, et al., "Efficient Path Selection for Delay Testing Based on Partial Path Evaluation", Proc. of 16th IEEE VLSI Test Symposium, pp. 188 - 193, 1998.
11. T. Haniotakis, Y. Tsiatouhas and D. Nikolos, "C-Testable One-Dimensional ILAs with Respect to Path Delay Faults : Theory and Applications", 1998 IEEE Int. Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 155 - 163.
12. C. Wu and T. Feng, "On a Class of Multistage Interconnection Networks", IEEE Transactions on Computers, vol. C-29 (8), pp. 694 – 702, August 1980.
13. M. A. Franklin, D.F. Wann and W.J. Thomas, "Pin Limitations and partitioning of VLSI Interconnection Networks", IEEE Trans. on Computers, vol. C-31, pp. 1109 – 1116, November 1982.
14. C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits", IEEE Trans. on CAD, pp. 694 – 703, September 1987.
15. K. Pramanick and S. M. Reddy, "On the Design of Path Delay Fault Testable Combinational Circuits", Proc. of Fault Tolerant Computing, pp. 374 – 381, 1990.
16. J. Lin, S. M. Reddy and S. Patil, "An Automatic Test Pattern Generator for the Detection of Path Delay Faults", Proc. of Int'l Conf. on CAD, pp. 284 – 287, 1987.
17. W. Ke and P. R. Menon, "Synthesis of Delay – Verifiable Combinational Circuits", IEEE Trans. on Computers, pp. 213 – 222, Feb. 1995.
18. H. S. Stone, "Parallel Processing with the perfect shuffle", IEEE Trans. on Computers, vol.C-20, pp. 153 - 161, 1971.