01 Jun 2004

Testing Layered Interconnection Networks

Bin Liu

Fabrizio Lombardi

Nohpill Park

Minsu Choi
Missouri University of Science and Technology, choim@mst.edu

Recommended Citation

B. Liu et al., "Testing Layered Interconnection Networks," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 710-722, Institute of Electrical and Electronics Engineers (IEEE), Jun 2004.
The definitive version is available at https://doi.org/10.1109/TC.2004.17

Testing Layered Interconnection Networks

Bin Liu, *Student Member*, *IEEE*, Fabrizio Lombardi, *Senior Member*, *IEEE*,
Nohpill Park, *Member*, *IEEE*, and Minsu Choi, *Member*, *IEEE*

Abstract—This paper presents an approach for fault detection in layered interconnection networks (LINs). An LIN is a generalized multistage interconnection network commonly used in reconfigurable systems; the nets (links) are arranged in sets (referred to as layers) of different size. Switching elements (made of simple switches such as transmission-gate-like devices) are arranged in a cascade to connect pairs of layers. The switching elements of an LIN have the same number of switches, but the switching patterns may not be uniform. A comprehensive fault model for the nets and switches is assumed at physical and behavioral levels. Testing requires configuring the LIN multiple times. Using a graph approach, it is proven that the minimal set of configurations corresponds to the node disjoint path sets. The proposed approach is based on two novel results in the execution of the network flow algorithm to find node disjoint path sets, while retaining optimality in the number of configurations. These objectives are accomplished by finding a feasible flow such that the maximal degree can be iteratively decreased, while guaranteeing the existence of an appropriate circulation. Net adjacencies are also tested for possible bridge faults (shorts). To account for 100 percent fault coverage of bridge faults a postprocessing algorithm may be required; bounds on its complexity are provided. The execution complexity of the proposed approach (inclusive of test vector generation and post-processing) is $O(N^4WL)$, where N is the total number of nets, W is the number of switches per switching element, and L is the number of layers. Extensive simulation results are provided.

Index Terms—Fault detection, layered interconnection networks, switch, fault tolerance, network flow.

1 Introduction

The efficient design and manufacturing of today's digital systems require programmability (accomplished through devices such as FPGAs, field programmable gate arrays) [16] and high connectivity (accomplished through complex high-speed interconnection networks such as crossbars and MINs, multistage interconnection networks) [4], [6]. Implementation of different types of multistage interconnection networks, together with their topological equivalence, can be found in [19], [24]. For example, point-to-point connections can be provided by using baseline networks [20]. Sparse interconnection networks consisting of cascading different networks have been proposed for configurable computing systems [1], [2].

High-performance computing has undergone a substantial change with the advent of configurable systems. These systems rely on interconnection resources such as nets and switches for programming flexibility, reduction in interchip communication and efficient routing [2], [4], [15]. The occurrence of faults in the nets and switches poses major concerns for continued system operation. Previous works on testing have dealt with different types of interconnection networks such as passive PCB-like networks [9], [12] and Multistage Interconnection Networks (MINs) [6], [7], [8]. Testing is usually accomplished by connecting the nets and setting the switches; the responses received at the outputs are then compared with the ones provided at the inputs. However, the presence of switches in an interconnection network substantially complicates the testing process [7]. The most serious limitation of current testing methods is that a homogeneous multistage structure is commonly assumed once switching elements are considered [6], [22]. This is not always applicable to practical systems as reconfiguration usually requires an irregular pattern [1], [4], [15]. The inability of current testing methods to deal with generic switching interconnection networks is further aggravated by the lack of a systematic testing algorithm [22].

A current trend in reconfigurable chip and system design is to have complex switching resources in the interconnection network [4] such that point-to-point switching and routing can also be accomplished between chips [15]. A layered interconnection network (LIN) has been proposed for reconfigurable systems [4] due to simplicity of implementation and flexibility in routing [1]. A LIN can be thought as an inhomogeneous MIN; the nets (links) are arranged in layers of different cardinality to connect the chips. The net layers are connected in cascade using switching elements. Switching elements have the same number of switches, but their pattern may not be uniform.

The objective of this paper is to propose a practical and structured approach to test layered interconnection networks (LINs) at production time under multiple faults. A divide-and-conquer approach is proposed under a comprehensive fault model. Given an LIN and the layout information (represented by an adjacency graph) for the nets, the proposed algorithm establishes the minimal number of configurations for testing the LIN. This algorithm is based on the iterative execution of the maximum

- *B. Liu was with the Department of Computer Science, Texas A&M University, College Station, Texas.*
- *F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115. E-mail: lombardi@ece.neu.edu.*
- *N. Park is with the Department of Computer Science, Oklahoma State University, Stillwater, OK 74078-1053. E-mail: npark@a.cs.okstate.edu.*
- *M. Choi is with the Department of Electrical and Computer Engineering, University of Missouri-Rolla, Rolla, MO 65409-0040. E-mail: choim@umr.edu.*

Manuscript received 20 Nov. 2002; revised 16 Sept. 2003; accepted 17 Sept. 2003.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 117799.

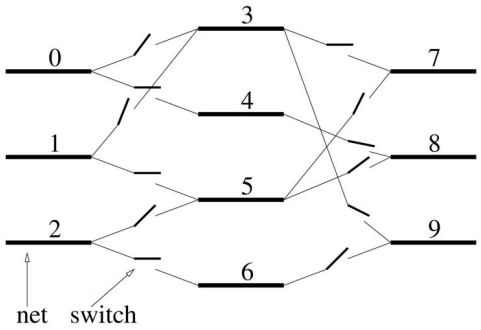

Fig. 1. An LIN with three layers.

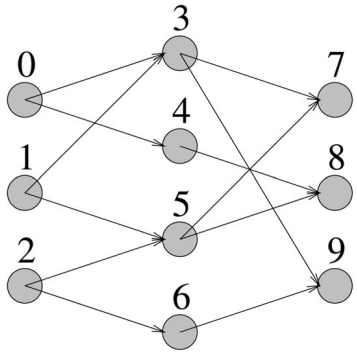

Fig. 2. The LIN graph G.

flow algorithm. It is proven that, by constructing different graphs, this process can be achieved optimally. A coloring algorithm is used to generate the test vectors used in each configuration. A postprocessing algorithm is used to deal with any remaining net adjacencies for detecting bridge faults. The novelty of this approach is that it has general applicability at polynomial time complexity, while attaining 100 percent fault coverage.

2 Preliminaries

An LIN consists of nets and switching elements. The nets (links) are arranged in L layers; each layer can have a different number of nets. Layers are connected in cascade using $L-1$ switching elements; some of the nets may not propagate to the next layer as they are used as local outputs. Similarly, local inputs can be provided to the internal switching elements. Each switching element consists of a fixed number of simple switches. Each switch connects a pair of nets in two adjacent and consecutive layers (note that the switches are unidirectional, i.e., not bidirectional); if a switch is ON (OFF), then the two nets are logically connected (disconnected). The switching patterns of the switching elements can be nonuniform. Without loss of generality and correctness, it is assumed that the primary (nonlocal) input and the output pins of the LIN are located on opposite sides. Fig. 1 shows an LIN with $L=3$; each switching element consists of six switches. Nets 0, 1, and 2 (7, 8, 9) are connected to the input (output) pins. All internal nets propagate through the layers (i.e., no internal net is a local output or input). This corresponds to the worst-case scenario due to the limited controllability/observability of the nets. The graph representation of an LIN is given by the directed acyclic graph $G=(V,E)$ (Fig. 2): In G, each node $v_i \in V$ represents a net; each edge $e(v_i,v_j) \in E$ represents the switch between the nets v_i and $v_j \in V$. Note that an edge is directed from L_i to L_{i+1} to represent a switch with a unidirectional and forward interconnection, where L_i represents the ith layer and is not the terminal layer. Also, it is assumed that each switch (i.e., edge) connects two adjacent and consecutive layers. To represent the connectivity of each node, the *degree* of a node is defined as equal to the outdegree of a node if it is not zero; or indegree, otherwise. Also, note that the definition of the degree is only valid for this particular type of networks, i.e., the indegree and outdegree of a node are equal when none of them is zero. If, in G, the $|V|=N$ nodes are arranged over L ($L \geq 2$) layers, then V can be expressed as a function of the layers as $V = V_{IN} \cup V_{OUT} \cup V_1 \cup V_2 \cup \ldots \cup V_{L-2}$, where, in the input (terminal) layer $V_{IN} = \{v_i^{IN}\}$, all nodes have an indegree (the number of input edges) equal to zero (a node v_i^{IN} in V_{IN} is called an input node); in the output (terminal) layer $V_{OUT} = \{v_i^{OUT}\}$, all nodes have an outdegree (the number of output edges) equal to zero (equivalently, a node v_i^{OUT} in V_{OUT} is called an output node); for all other internal layers $V_j = \{v_i^j\}$ $(1 \leq j \leq L-2)$, the indegree of each node is equal to its outdegree (and a node v_i^j in the internal layer V_j is called an internal node). The maximal degree D_p of G is the maximal degree of all nodes, i.e., $D_p = MAX(d_1^p, d_2^p, \ldots, d_N^p)$, where d_i^p $(1 \leq i \leq N)$ is the degree of node v_i. The first layer is V_{IN} and the last layer is V_{OUT}.

Note that an LIN is a generalization of an MIN; for example, a K-stage MIN [24] (such as a delta network) with switches with two inputs and two outputs is topologically equivalent to an LIN with $N=LK$, $L-1=log_2 K$ and $2K=W$ (the switching patterns of the elements are arranged accordingly, such as the perfect shuffle [23], [24]).

The above graph model can be extended with no loss of correctness to the case of internal or local input/output nets. The following features are assumed with respect to the operation and structure of the LIN:

1. One-to-one switching and routing are assumed.
2. The layout of the nets of the LIN is known.
3. The state of each switch can be individually set (either ON or OFF) using an appropriate mechanism.
4. The logically disconnected output value of a switch is assumed to be 0 (nonfloating state) [21].

The layout information is represented by an undirected graph $G_{ad} = (V_{ad}, E_{ad})$ referred to as the adjacency graph. Every node in V is also a node in V_{ad}, i.e., $V = V_{ad}$. If two nodes v_i and v_j are adjacent in the physical layout and a bridge fault can exist [5], then there is an edge $e_{ad}(v_i,v_j) \in E_{ad}$. It is assumed that, for a bridge to exist, there is no constraint on the involved two nets to be in two different disjoint paths and two adjacent nets to be in the same layer. The degree D_{ad} is the maximal degree of all nodes in V_{ad} of G_{ad}, i.e., $D_{ad} = MAX(d_1^{ad}, d_2^{ad}, \ldots, d_N^{ad})$, where d_i^{ad} $(1 \leq i \leq N)$ is the degree of node v_i^{ad}. Fig. 3 shows a possible (randomly generated) G_{ad} of G in Fig. 2. No restriction is assumed on the adjacencies in G_{ad}; for

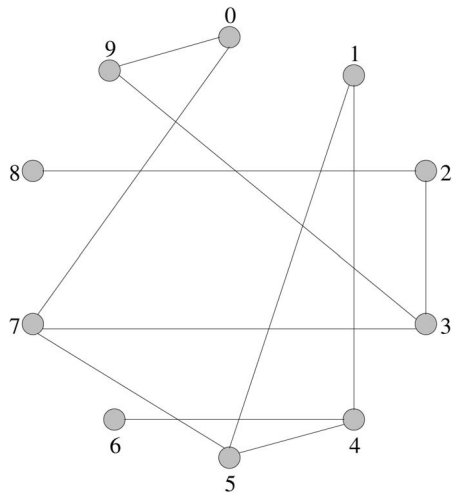

Fig. 3. Adjacency graph G_{ad} of G.

example, v_0 and v_9 (v_4 and v_5) are connected in Fig. 3; they are in the first and third (same) layers, respectively.

Two types of faults are considered: net faults and switch faults. Net faults are defined as follows in the graph representation:

1. For each vertex (net) v_i in G: stuck-at-1, stuck-at-0.
2. For each pair of vertices (nets) v_i and v_j connected in G_{ad} by an edge: short (bridge).

For a switch fault, a behavioral model is assumed as in previous papers [6], [7], [8], [22], i.e., testing a switch is equivalent to testing its ON state to connect two nets in the layers. No restriction is placed on the number of faults existing in the LIN. Hereafter, the following additional assumptions are valid in the analysis.

1. A structural bridge fault model based on G_{ad} is applicable to the shorts. The OR short is assumed for simplicity. This is an assumption that affects only the test vector generation process [5].
2. Probing is only allowed at the input and output pins of the LIN.
3. $D_{ad} \ll N$, i.e., G_{ad} is relatively sparse (as commonly found in practice).

For compatibility with existing literature, the following terminology is used [5], [22] (included for completeness):

1. *Node Disjoint Path (NDP)*: Two node sets, S_p and S_q (consisting of nodes from the input layer to the output layer), are said to be disjoint if they have no node in common, i.e., $S_p \cap S_q = \emptyset$. The paths which cover the nodes in each set and the edges between the nodes make up the node disjoint paths.
2. *Configuration*: This is a set of paths from the input to the output pins by programming the switches.

3 Review

This section presents a brief review of behavioral and structural testing approaches for wiring interconnects as well as interconnection networks.

No switch is present in a wiring interconnect. In behavioral testing, it is commonly postulated that every net can be shorted to any other net, so the *Counting Sequence Algorithm* of [12] can be used to detect all bridge faults with a test length of $\lceil \log_2 N \rceil$, where N is the number of nets. A modified counting sequence of length $\lceil \log_2(N+2) \rceil$ can be used to detect stuck-at faults too. The *walking-1* test set proposed by [18] can be used to diagnose all bridge faults. The drawback to this method is that large test sets are generated as a sequence of length N is required; however, test generation is simple.

For structural testing, the wiring layout of the network must be specified. Chen and Lombardi [5] have proposed a new structural approach. This approach utilizes different graph coloring techniques and appropriate codes to generate a test set; [5] has shown that graph coloring is an efficient technique for interconnect testing by utilizing a method which relates the size of the shorts and the adjacency information of the interconnect to the codes for the tests. First, a simple graph technique is utilized for color generation and fault detection. By simulation, [5] has also shown that, for benchmarks and random interconnects, this approach requires a smaller number of tests than previous approaches; it has also been proven that its time complexity is $O(max\{N^2, N \times D_{ad}^3\})$. This structural approach generates a test set whose cardinality is upper bounded by $2D_{ad} + 2$.

Testing of interconnection networks such as MINs has been extensively analyzed [6], [7], [8], [22] either for online and/or manufacturing testing [7], [21]. It has been proven [6] that, under a single fault assumption in a behavioral model (based on the combinatorial analysis of the switching states), a baseline interconnection network with $log_2 K$ stages (each stage consists of $K/2$ switching elements, each switching element consists of two input and two output nets) can be diagnosed using $max\{12, 6 + 2\lceil log_2(log_2 K) \rceil\}$ vectors, where K is the number of input (output) pins of the MIN.

Also, it has been proven [8] that a baseline interconnection network made of switching elements with two input and two output nets can be diagnosed under a single fault assumption and the behavioral model of [6] using only 12 tests, i.e., in a testing complexity independent of network size [7]. A similar result has been also obtained in [22]: Under a functional fault model (consisting of conflict resolution and data multiplexing faults) for the operation of the MIN and a stuck-at and bridge fault model for the control lines and nets, [22] has proven that $f \times f$ switches in a homogeneous MIN can be tested using a number of test vectors independent of size (the complexity for diagnosing the MIN is still $O(log_2 K)$).

Few papers have addressed multiple fault diagnosis of MINs [6], [8]. Multiple fault diagnosis is needed for manufacturing. In [6], it has been proven that $2(1 + log_2 K)$ tests may detect, in most cases, multiple faults under a full combinatorial fault model for the switching elements and a stuck-at model for the nets. The MIN must be configured $1 + log_2 K$ times. Feng et al. [8] have presented an algorithm for multiple fault diagnosis in interconnection networks made of baseline switching elements. This approach requires the execution of $(1 + log_2 K)$ phases, each made of two tests. It is applicable provided that there exist no

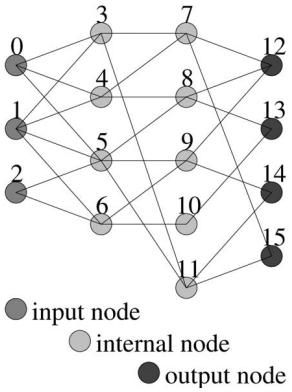

Fig. 4. Example of LIN graph G with four layers.

logically erroneous outputs or no logically unidentified outputs in the faulty switching elements; also, multiple link faults (of the stuck-at-0 or stuck-at-1 type) are assumed not to exist in the network.

4 Maximum Flow

In this section, two novel fundamental results as applicable to the maximum flow [10] will be presented. The maximum flow algorithm is based on the following theorem, generally known as Hoffman's Theorem [11], [13].

Theorem 1 (The Circulation or Hoffman's Theorem). *In a network $G = (V, E)$ with lower edge capacity bounds $B_l = \{l(v_i, v_j)\}$ and upper edge capacity bounds $B_c = \{c(v_i, v_j)\}$, a feasible integer circulation exists if and only if $\sum_{i\in S, j\in T} l(v_i, v_j) \le \sum_{i\in S, j\in T} c(v_j, v_i)$ for every cutset $S : T$, where $T = V - S$.*

The following new theorem proves a fundamental property of an LIN.

Theorem 2 (Node disjoint Paths Theorem). *For an LIN graph $G = (V, E)$ with maximal degree D_p, it is possible to find a set of node-disjoint paths (from the input layer to the output layer) which cover all nodes of degree D_p. Or, alternatively, for an LIN graph G with maximal degree D_p, it is possible to find a flow to cover all nodes with degree D_p.*

Proof. This theorem will be proven by construction of different graphs. First, add edges $\{e(v_i^{OUT}, v_j^{IN})\}$ from V_{OUT} to V_{IN} such that, for every node, the indegree is equal to the outdegree. Then, split each original node v_i into two nodes (denoted as v'_{i1} and v'_{i2}) and place an edge between v'_{i1} and v'_{i2}. Each original edge $e(v_i, v_j)$ is replaced by a new edge $e(v'_{i2}, v'_{j1})$ in $G' = (V', E')$. So, for example, the graph G in Fig. 4 is transformed into G' of Fig. 5. An edge in G' is given as follows:

1. *A return edge* $e'_r(v'^{OUT}_{i2}, v'^{IN}_{j1})$, if it connects an output node V'^{OUT}_{i2} to an input node V'^{IN}_{j1}; its lower capacity is $l(v'^{OUT}_{i2}, v'^{IN}_{j1}) = 0$, while the upper capacity is $c(v'^{OUT}_{i2}, v'^{IN}_{j1}) = \infty$.
2. *A path edge* $e'_p(v'_{i2}, v'_{j1})$, if this edge exists in the original network G as $e(v_i, v_j)$; its lower capacity is $l(v'_{i2}, v'_{j1}) = 0$, while its upper capacity is $c(v'_{i2}, v'_{j1}) = \infty$.

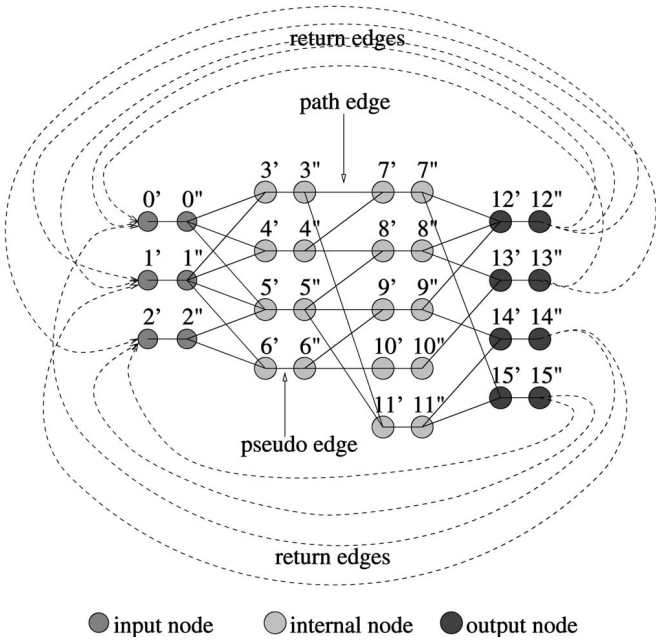

Fig. 5. G' constructed from G.

3. *A pseudoedge* $e'_s(v'_{i1}, v'_{i2})$, if this is the edge between the two split nodes v'_{i1} and v'_{i2} of the original node v_i; its upper capacity is $c(v'_{i1}, v'_{i2}) = 1$, while its lower capacity is $l(v'_{i1}, v'_{i2}) = 1$ (provided the node has maximal degree), otherwise $l(v'_{i1}, v'_{i2}) = 0$.

As $V' = \{v'_{i1}, v'_{i2}\}$;

$$E' = \{e'_s(v'_{i1}, v'_{i2})\} \cup \{e'_p(v'_{i2}, v'_{j1})\} \cup \{e'_r(v'^{OUT}_{i2}, v'^{IN}_{j1})\},$$

and $\{e'_r(v'^{OUT}_{i2}, v'^{IN}_{j1})\}$ satisfies the definition of LIN as given previously.

We are interested in proving that a circulation exists in G'. Based on Hoffman's Theorem, such a circulation exists if and only if, for any cutset, the sum of the lower capacities of the forward edges across each cut does not exceed the sum of the upper capacities of the backward edges across the cut (in this case, a cut is a partition of the nodes into two sets, given by S and $T = V - S$). The forward edges are the edges from S to T, while the backward edges are the edges from T to S. Also, note that, in Fig. 5, the indegree of each node in the first layer as circulations back from the last layer is arbitrary and independent of the outdegree of each node in the first layer.

In Fig. 6, a node v_i of the original network G can be denoted as follows:

1. *An entry node*, if v'_{i1} is in S, v'_{i2} is in T (for example, node v).
2. *An exit node*, if v'_{i1} is in T, v'_{i2} is in S (for example, node u).
3. *An inner node*, if v'_{i1} is in T, v'_{i2} is in T (for example, node p).
4. *An outer node*, if v'_{i1} is in S, v'_{i2} is in S (for example, node q).

So, it is also possible to designate all edges of G' as return, path, and pseudoedges and all nodes of the original network G as entry, exit, inner, or outer nodes.

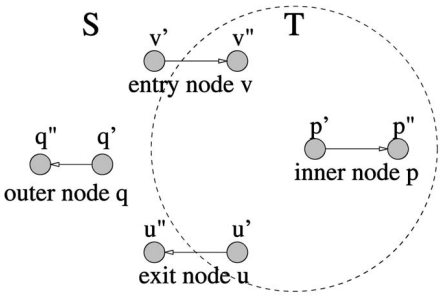

Fig. 6. Definition of entry, exit, inner, and outer nodes.

The following statements are valid: 1) The sum of the lower capacities of the forward edges is given by A, where A is the number of entry nodes with maximal degree. 2) The sum of the upper capacities of the backward edges across the cut is given by B, where B is the number of exit nodes provided the original network G contains no edge from an entry (or inner node) to an entry (or outer node); otherwise, B is given by infinity (i.e., if the cut makes at least a return or path edge as the backward edge, then the sum of the upper capacities of the backward edges is equal to infinity).

One of the following three scenarios must occur for any cutset $S:T$: 1) No entry node has maximal degree. In this case, the sum of the lower capacities of the forward edges is zero. So, Hoffman's Theorem is satisfied. 2) The set of backward edges includes at least one return edge or path edge. In this case, the sum of the upper capacities of the backward edges is infinity. So, Hoffman's Theorem is satisfied. 3) The cut only crosses the pseudoedges of the entry and exit nodes. Let O denote the sum of the outdegrees of the entry nodes and I denote the sum of the indegrees of the exit nodes. Then, $O = I$ (based on the features of the graph). If the maximal degree of the network is D_p, then $D_pA \leq O$ and $I \leq D_pB$, or $D_pA \leq D_pB$, i.e., $A \leq B$. Hoffman's Theorem is also satisfied.

From the above results, for any cutset $S:T$, Hoffman's Theorem is always satisfied. Therefore, it is possible to always find a feasible flow to cover all nodes with maximal degree. □

As per the above theorem, the next theorem is applicable for testing an LIN.

Theorem 3 (Network Decomposition Theorem). *For an LIN graph $G = (V, E)$ with maximal degree D_p, it is always possible to decompose G into D_p configurations (or NDP sets); these sets cover all nodes and edges of the graph.*

Proof. Based on Theorem 2, in a graph with maximal degree D_p, we can find the NDP set (from the input layer to the output layer) to cover all nodes with degree D_p. After finding this set, all edges on these paths are removed and then a new graph of maximal degree $D_p - 1$ is obtained. This graph still satisfies the network definition. By repeating this procedure, D_p NDP sets (from the input layer to the output layer) are obtained to cover all nodes (with reuse) and edges (without reuse) in G. □

5 Test Generation Procedure

A divide-and-conquer procedure is employed. Given the graph G of the LIN and an adjacency graph G_{ad}, the proposed approach consists of two parts.

1. Generation of LIN configurations by finding the NDP sets.
2. Test vector generation.

The proposed approach is based on the equivalence between LIN testing and the node-disjoint path (NDP) sets problem. The NDP sets problem can be stated as follows: Given an acyclic directed graph G, find the minimal number of sets (each set made of disjoint paths from the input to the output vertices) such that an edge in G is in at least a node disjoint path set and no node is shared between paths in each set. This is the simplest instance of equivalence between these two problems because LIN testing is also constrained by covering all edges in G_{ad} (if bridge detection is required). The minimal number of NDP sets corresponds to the minimal number of configurations of the LIN; in each phase, the LIN is configured by turning ON the appropriate switches and establishing the desired connections for the node disjoint paths.

5.1 Part 1: Generation of LIN Configurations

In the first part, the graph G is decomposed into a minimal number of subgraphs ($\{G_i\}$ denotes the set of the subgraphs) by setting the switches (given by the edges in E) as either open (OFF) or closed (ON). Each subgraph corresponds to an NDP set (and an LIN configuration). By considering all G_i, the corresponding subgraphs $\{G_i^{ad}\}$ are generated from the original G_{ad}. $\{G_i^{ad}\}$ is generated from G_{ad} to satisfy the new interconnection structure of each $\{G_i\}$. If some net adjacencies are not covered by $\{G_i^{ad}\}$, a residual adjacency graph G_R^{ad} (consisting of all remaining adjacencies) is used for postprocessing (note that the details about postprocessing residual adjacency graph are explained at the end of this section, and a pseudocode can be found in Appendix B). The number of subgraphs (Z) must be as small as possible because this determines the minimal number of configurations.

This process can be expressed as

$$G(V, E) = \bigcup_{i=1}^{Z} G_i(V_i, E_i),\ E = \bigcup_{i=1}^{Z} E_i,$$

where $E_i \cap E_j = \emptyset$ for $i, j = 1, \ldots, Z$;

$$G_{ad}(V_{ad}, E_{ad}) = \bigcup_{i=1}^{Z} G_i^{ad} + G_R^{ad},\ E_{ad} = \bigcup_{i=1}^{Z} E_i^{ad} \cup E_R^{ad},$$

where $E_i^{ad} \cap E_j^{ad} = \emptyset$ for $i, j = 1, \ldots, Z,$ or R.

The proposed algorithm tests a configuration in each phase. As one-to-one switching and routing are assumed and using the Network Decomposition Theorem (Theorem 3), the lower bound in the number of configurations (and, equivalently, phases) is given by the maximal degree of G (D_p). The proposed algorithm decomposes G into $Z = D_p$ subgraphs by using the maximum flow algorithm with upper and lower capacity bounds. Once the configurations

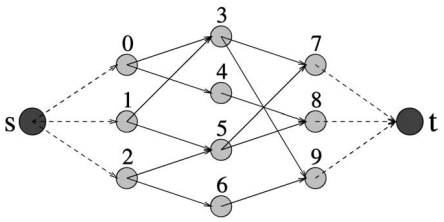

Fig. 7. Construction of G'_1.

are found, then a test vector generator is used, i.e., testing of the LIN consists of testing each configuration at a time.

It is not possible to immediately use the maximum flow algorithm on G (as in Fig. 2) because this graph may have multiple sources (inputs) and sinks (outputs). Two vertices are therefore added: a supersource (denoted by s) and a supersink (denoted by t). Also, directed edges from s to each original source node and from each original sink node to t are added. This new graph converts a multiple sources (sinks) network G into a single source (sink) flow network $G'_1 = (V'_1, E'_1)$ $(G \rightarrow G'_1)$, where $V'_1 = V \cup \{s, t\}$; $E'_1 = E \cup \{e'_1(s, v) | v \in V_{IN}\} \cup \{e'_1(v, t) | v \in V_{OUT}\}$. The maximal degree of a graph in each phase must also be considered. The flow must be directed to pass through all the nodes with maximal degree so that the already traversed paths are not considered in the next phases, i.e., to decrease the maximal degree of the graph in each phase. Therefore, the maximum flow approach with upper and lower capacity bounds [10] is employed. To calculate the bounded maximum flow, a new graph must be constructed from G'_1. Assume that each edge $e(u, v)$ in $G = (V, E)$ has an upper capacity bound $c(u, v)$ on the net flow from u to v, and a lower capacity bound $l(u, v)$, i.e., any flow f on the network must satisfy $l(u, v) \leq f(u, v) \leq c(u, v)$. Let s and t be the source and sink of G'_1. Construct from G'_1 the flow network $G'_2 = (V'_2, E'_2)$ with upper capacity bound c', source s', and sink t' as follows $(G'_1 \rightarrow G'_2)$: $V'_2 = V'_1 \cup \{s', t'\}$;

$$E'_2 = E'_1 \cup \{e'_2(s', v) : v \in V'_1\} \cup \{e'_2(u, t') : u \in V'_1\} \cup \{e'_2(s, t), e'_2(t, s)\}.$$

For example, the flow graph shown in Fig. 8 is obtained from the graph in Fig. 7.

To guarantee that a feasible flow will traverse all nodes with maximal degree, each node v_i (except s, t, s', and t') is split into two pseudonodes (v'_{i1} and v'_{i2}). A feasible flow may not traverse all nodes with maximal degree with no pseudonodes; pseudonodes guarantee disjoint paths to traverse adjacent layers using the max flow algorithm. intact. A pseudoedge $e'(v'_{i1}, v'_{i2})$ is added between them

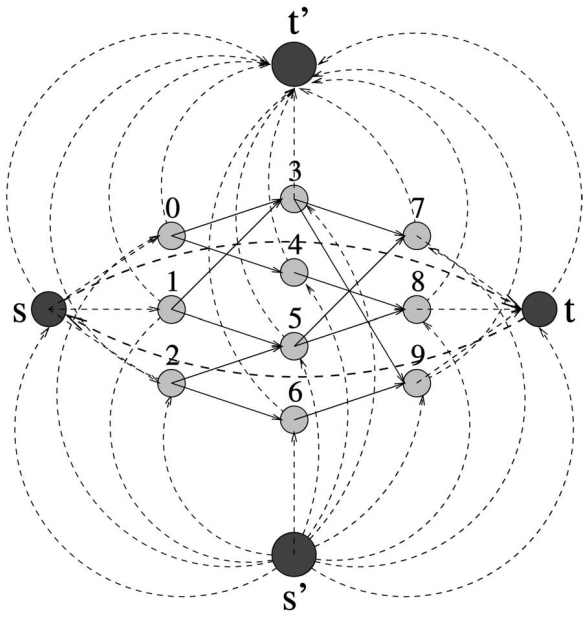

Fig. 8. Construction of G'_2.

(Fig. 9). After considering the capacity bounds, the final auxiliary graph $G' = (V', E')$ (Fig. 10) is constructed $(G'_2 \rightarrow G')$ such that $V' = \{v'_{i1}, v'_{i2}\} \cup \{s, t\} \cup \{s', t'\}$;

$$E' = \{e'(v'_{i1}, v'_{i2})\} \cup \{e'(s', v'_{i2})\} \cup \{e'(v'_{i1}, t')\} \cup \{e'(s, t), e'(t, s)\}.$$

For the flow to move along the expected paths, the upper capacities (denoted as c for G or c' for G') and lower capacity (denoted as l for G) of the edges are given as follows:

1. For each edge $e'(u, v) \in E'$,

$$c(u, v) = 1;$$
$$l(u, v) = \begin{cases} 1 & \text{if } e'(u, v) \text{ is a pseudoedge with maximal degree,} \\ 0 & \text{otherwise;} \end{cases}$$
$$c'(u, v) = c(u, v) - l(u, v).$$

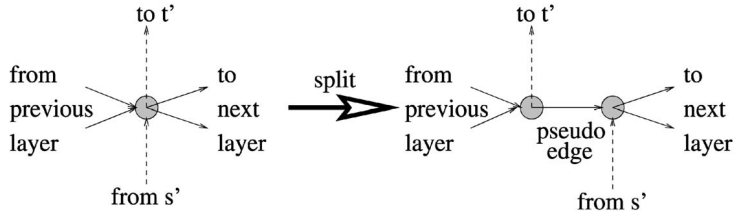

Fig. 9. Node splitting to set the flow path.

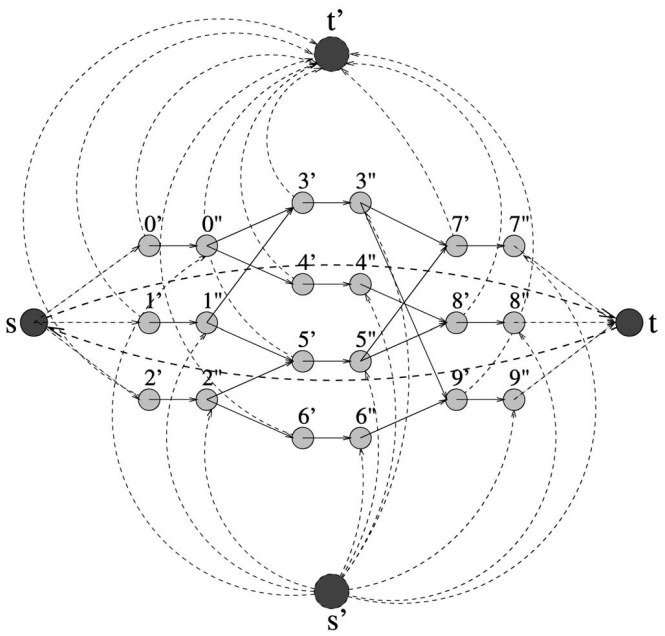

Fig. 10. Construction of the final auxiliary graph G'.

2. For each node $u \in V'$,

$$c'(s', u) = \sum_{v' \in V'}^{all} l(v', u);$$

$$c'(u, t') = \sum_{v' \in V'}^{all} l(u, v').$$

3. $c'(s, t) = c'(t, s) = \infty$.

The next step consists of finding the feasible flow on the final auxiliary graph G' using the Preflow-Push algorithm [10]. The pseudocode of this step is given in Appendix A. Based on the network decomposition theorem (Theorem 3), it is possible to always find a feasible flow in the LIN.

The Preflow-Push algorithm [10] has a complexity of $O(|V'|^2|E'|)$ for a G' with $|V'|$ nodes and $|E'|$ edges. The LIN graph G has L layers and W switches per switching element, so there are a total of $W(L-1)$ switches (i.e., $W(L-1)$ edges in G). Assume that the LIN has N nets (i.e., N nodes in G), then each node is split into two nodes and three more edges are added to the graph. So, the total number of edges in G' for processing by the Preflow-Push algorithm is $W(L-1)+3N+6$ and the total number of nodes is $2N+4$. Therefore, the computational complexity for each feasible flow search is $O(N^2WL)$ for $WL \gg N$. For decomposing the LIN graph into subgraphs, the maximum flow algorithm must be executed D_p times, then its computational complexity is $O(D_pN^2WL)$.

After decomposing the LIN graph into subgraphs and finding the configurations, the maximum flow algorithm is used for processing the residual adjacency graph (if detection of bridge faults is required). By setting to 1 the lower capacity of each net with a residual adjacency, it is then possible to find a feasible flow to cover as many nets with remaining adjacencies as possible. The pseudocode of the residual adjacency graph processing algorithm can be found in Appendix B. The residual adjacency graph must be considered M times (i.e., M phases to cover all residual adjacencies, bounds on M are given in a later section), then the computational complexity for residual graph processing is $O(MN^2WL)$. So, the overall complexity for Part 1 is $O(\max(M, D_p)N^2WL)$. Note that if G has disjoint components, then the proposed approach must be applied to all components and they can be tested in parallel.

5.2 Part 2: Test Vector Generation

In the second part, a test generator is used to obtain the vectors for testing a LIN configuration in a phase. Many structural algorithms have been proposed in the literature [5]. For simplicity, the algorithm of [5], referred to as simple coloring, is used for test vector generation and fault detection. Fault detection using simple coloring is based on a greedy condition in the adjacency graph: The node with the highest degree is selected first using the lowest number of possible colors. This coloring technique [5] achieves a considerable reduction in the number of test vectors compared with previous approaches based on structural information; the reduction is more pronounced when D_{ad} is much smaller than N. However, simple coloring does not guarantee full diagnosis (diagnosis requires a more complex coloring of the graph with a corresponding increase in phases and tests). The computational complexity of the simple coloring algorithm is $O(N^2 + D_{ad} \times N + D_{ad} \times N) = O(N^2)$ (for $N \gg D_{ad}$), which implies that the simple coloring algorithm adds no further complexity to the process of generating the adjacency graph. Note that if diagnosis is the desired objective, more sophisticated algorithms must be employed for test generation based on coloring [5].

The fault coverage of the proposed testing process is given as follows:

1. At completion of all configurations, every switch in the LIN is part of at least a path (i.e., it should be ON at least once); under the behavioral model [6], [7], if a switch is incorrectly OFF, then the selected path is disconnected. These faults are detected.
2. All adjacencies in G_{ad} are dealt using either the Network Decomposition Algorithm (Appendix A) or the Residual Graph Processing Algorithm (Appendix B). Vectors are generated using graph coloring [5]; hence, 100 percent coverage of all bridge faults is achieved.
3. Every net is provided with a 0 and a 1 as inputs; hence, all stuck-at faults are also detected.

Hence, 100 percent coverage of multiple faults in the switches and nets is accomplished using the proposed divide-and-conquer approach.

6 Complexity Analysis

Consider test generation. There are at most N_c paths in a configuration or NDP set (where N_c is the minimal number of nets in a layer), so there are at most N_c nodes in the adjacency graph. The complexity of fault detection by coloring each configuration is $O(N_c^2)$. For fault detection, it has been shown [5] that the cardinality of the test set

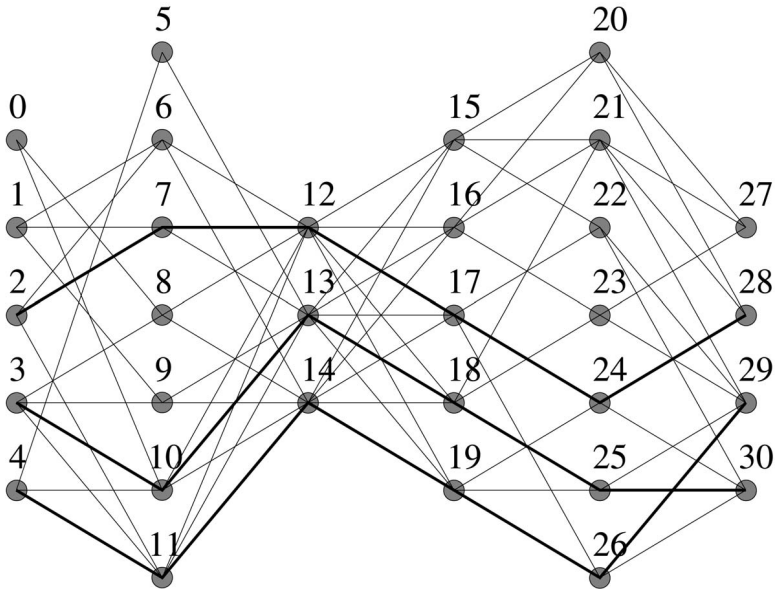

Fig. 11. Node-disjoint path (NDP) set, configuration 1.

generated using simple coloring [5] is no more than the modified counting sequence as the worst case for coding the colors. D_p configurations are needed to test the LIN and M configurations are needed for the residual adjacency graph (if required); so, a total of $(D_p + M)$ configurations must be tested. The overall time complexity is therefore $O((D_p + M)N_c^2)$.

Consider next the test complexity of the proposed approach. The total number of phases is the same as the number of configurations, i.e., $D_p + M$. Assuming a complete adjacency graph for a configuration (the same as in a behavioral case), the number of colors needed is at most N_c. This means that at most $\lceil log_2(N_c + 2)\rceil$ tests will be used for a configuration. Therefore, the total number of tests is $(D_p + M)\lceil log_2(N_c + 2)\rceil$. Note that D_p is determined by the structure of the LIN, while M is determined by both the structure of the LIN and the adjacency graph. Different adjacency graphs will result in different values of M.

A more comprehensive analysis must be pursued with respect to M. Consider the lower bound. As mentioned previously, the LIN graph $G = (V, E)$ is guaranteed to be decomposed into D_p NDP sets where D_p is the maximal degree of G, i.e., $G(V, E) = \{NDP_i = G_i(V_i, E_i)|1 \le i \le D_p\}$; $E = E_1 \cup E_2 \cup \ldots \cup E_{D_p}$, where $E_i \cap E_j = \emptyset$. As each NDP set corresponds to a configuration and each configuration is tested in a phase, then all switches are tested in D_p phases; however, not necessarily all adjacencies are tested in D_p phases as additional phases (whose number is given by M) may be needed to cover the remaining adjacencies. Therefore, the total number of phases (denoted by N_p) for the proposed approach to LIN testing is $(D_p + M)$. When $M = 0$, then the lower bound for the number of phases is $N_p = D_p$.

As for the upper bound, two cases must be distinguished, i.e., the worst and best cases. First, the definitions of the maximal size of a layer N_x and the minimal size of a layer N_c are introduced. For an LIN with L layers $G = (V, E)$, $V = V_{IN} \cup V_{OUT} \cup V_1 \cup V_2 \cup \ldots \cup V_{L-2}$. As $|V_i|$ represents the number of nodes in the ith layer, then

$$N_x = MAX(|V_{IN}|, |V_{OUT}|, |V_1|, \ldots, |V_{L-2}|);$$
$$N_c = MIN(|V_{IN}|, |V_{OUT}|, |V_1|, \ldots, |V_{L-2}|).$$

The layer with N_x nodes is referred to as layer V_x. The layer with N_c nodes is referred to as layer V_c.

The worst case occurs when the adjacency graph is a complete graph and the adjacencies are tested one by one. Therefore, the upper bound is given as

$$N_p = \binom{N}{2} = \frac{N(N-1)}{2} = O(N^2).$$

For an LIN with fully connected switching elements, a different upper bound can be established as follows: Divide the total adjacencies into two groups: $\{A_i\}$, the adjacencies of nets within the ith layer, and $\{A_{ij}\}$, the adjacencies between two different adjacent layers (i and $j = i + 1$). As $\{A_i\}$ of different layers can be tested in parallel, then $\{A_{ij}\}$ can be covered when $\{A_i\}$ are tested. Therefore, the maximal number of phases to test the adjacencies is $M = N_x(N_x - 1)/2$, i.e., the maximal number of adjacencies in the layer V_x. In this case, the upper bound is $N_p = D_p + N_x(N_x - 1)/2 = O(N_x^2)$.

The best case occurs when every layer has the same cardinality, i.e., $|V_{IN}| = |V_{OUT}| = |V_1| = \ldots = |V_{L-2}|$. In this case, it is possible to find at least one NDP set to cover all nodes in G (as there should be at least one NDP set from the inputs to the outputs). This corresponds to a partially homogeneous configuration for an LIN. Then, only one phase is required for testing all adjacencies. Thus, the upper bound is $N_p = D_p + 1$. As $M = O(N^2)$ and using [5], The residual adjacency graph processing part is $O(N^2)$, then the overall complexity of the proposed approach is $O(N^4WL)$.

TABLE 1
The Node (Net) Adjacencies for the LIN

node	adjacencies	node	adjacencies	node	adjacencies	node	adjacencies
0	16,29,30	8	10,17	16	0,4,10,12	24	9,10
1	2,20,28	9	24	17	8,15,30	25	12,15,30
2	1,19,23	10	8,16,21,24	18	5,11	26	7
3	NONE	11	5,18	19	2	27	NONE
4	16	12	7,16,25,30	20	1	28	1,7
5	11,18	13	NONE	21	7,10,15	29	0
6	NONE	14	NONE	22	NONE	30	0,12,17,25
7	12,21,26,28	15	17,21,25	23	2		

7 An Example

Consider the LIN graph G shown in Fig. 11. This is an example of an LIN with 6 layers, 31 nets, and 75 switches (15 switches per switching element). The maximal degree is $D_p = 5$. In this LIN, the number of input pins is not equal to the number of output pins. The number of nets per layer is also variable.

The adjacencies (and, therefore, G_{ad}) are generated randomly and are given in Table 1. In Part 1, the LIN is initially decomposed into five NDP sets (Set 1-5) using the algorithm in Appendix A. For example, the dark edges in Fig. 11 show the first NDP set. Two additional NDP sets (sets 6 and 7) are generated to cover the residual adjacencies using the algorithm in Appendix B. The NDP sets are given as follows (each path is identified by the nodes it traverses):

```
Set (Configuration) 1:  4 - 11 - 14 - 19 - 26 - 29
                        3 - 10 - 13 - 18 - 25 - 30
                        2 - 7 - 12 - 17 - 24 - 28
Set (Configuration) 2:  4 - 10 - 14 - 16 - 23 - 27
                        3 - 11 - 13 - 17 - 26 - 30
                        2 - 6 - 12 - 19 - 25 - 29
Set (Configuration) 3:  3 - 8 - 14 - 17 - 22 - 30
                        2 - 11 - 12 - 18 - 23 - 29
                        1 - 9 - 13 - 15 - 21 - 28
Set (Configuration) 4:  3 - 9 - 14 - 18 - 21 - 27
                        1 - 7 - 13 - 16 - 20 - 28
                        0 - 10 - 12 - 15 - 22 - 29
Set (Configuration) 5:  4 - 5 - 13 - 19 - 24 - 30
                        1 - 6 - 14 - 15 - 20 - 27
                        0 - 8 - 12 - 16 - 21 - 29
Set (Configuration) 6:  4 - 5 - 13 - 19 - 24 - 28
                        3 - 9 - 14 - 18 - 25 - 30
                        2 - 11 - 12 - 15 - 22 - 29
Set (Configuration) 7:  4 - 10 - 13 - 18 - 25 - 29
                        3 - 8 - 14 - 19 - 26 - 30
```

Using the simple coloring algorithm, in Part 2 we can find that Sets 2, 5, and 7 need only teo colors; all other sets need three colors. Then, using the modified counting sequence, we need two tests for Sets 2, 5, and 7 each and three tests for each of the other sets. Table 2 shows the tests applied sequentially to each primary input pin (the STVs for Nodes 0, 1, 2, 3, and 4) for all configurations (note that, in this table, "X" means "don't care," i.e., 0 for OR type bridge and 1 for AND type bridge).

Testing of this LIN requires seven configurations (or phases) and a total of 18 tests. Note that none of the previous approaches found in the literature [6], [7], [8], [22] can be used to test this interconnection network.

TABLE 2
Test Vectors for Each NDP Set

	Node 0	Node 1	Node 2	Node 3	Node 4
STV (Configuration 1)	XXX	XXX	101	110	011
STV (Configuration 2)	XX	XX	10	01	01
STV (Configuration 3)	XXX	101	110	011	XXX
STV (Configuration 4)	101	110	XXX	011	XXX
STV (Configuration 5)	10	01	XX	XX	01
STV (Configuration 6)	XXX	XXX	101	110	011
STV (Configuration 7)	XX	XX	XX	10	01

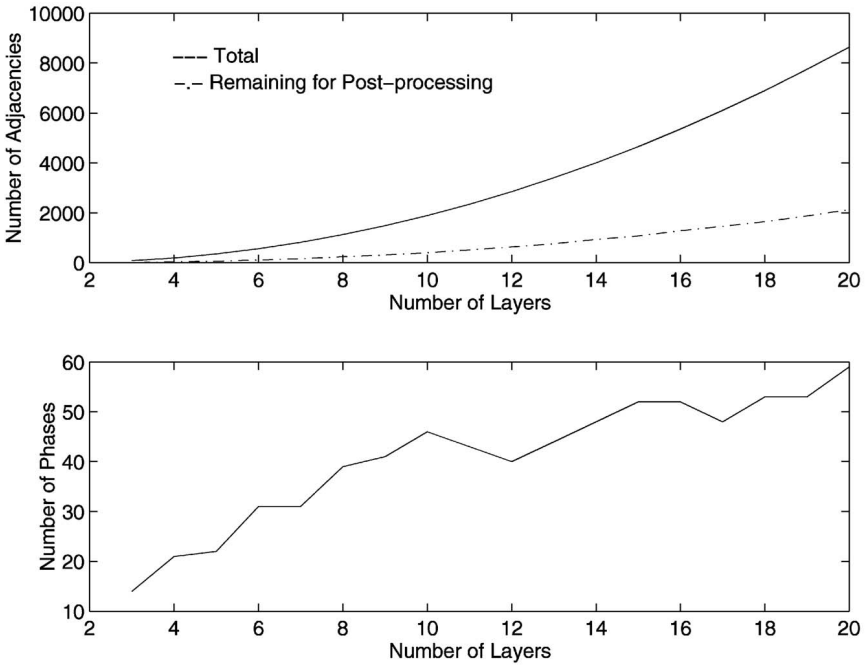

Fig. 12. Number of phases versus number of layers.

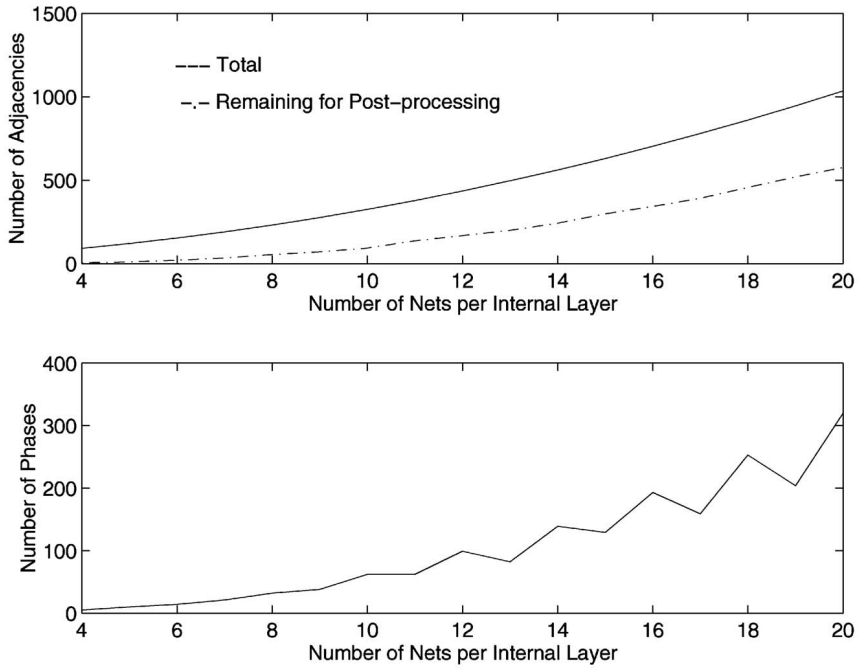

Fig. 13. Number of phases versus number of nets.

8 Simulation Results

Several experiments have been performed to evaluate the proposed approach. All LINs used in these experiments have input and output layers with three nets each and some random characteristics. All adjacency graphs are fully connected (the worst case scenario). Different experiments were conducted. Note that the number of adjacencies and phases were used as the primary figure of merits as they are the primary factors to determine the complexity of runtime of each algorithm.

1. The variable is the number of internal layers (i.e., $L-2$). The simulated LINs have seven nets per internal layer, and 15 switches per switching element. Fig. 12 shows the results. (Note that, in Figs. 12, 13, and 14, the numbers of total and remaining adjacencies for postprocessing are represented by solid and dotted lines, respectively.)
2. In this experiment, the variable is the numbers of nets in the internal layers. The simulated LINs have two internal layers and a 70 percent connectivity (i.e., 70 percent of the switches of a fully connected

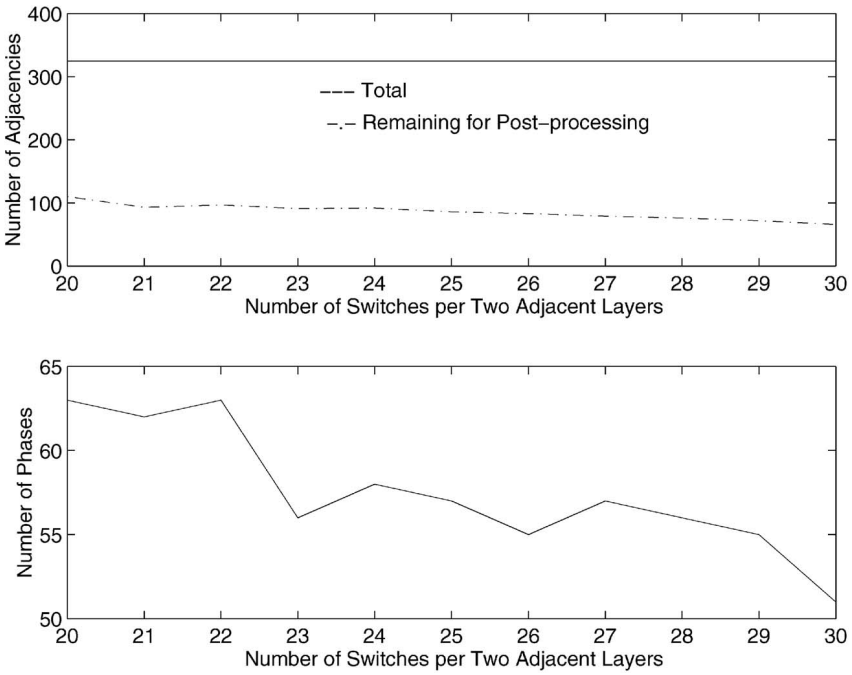

Fig. 14. Number of phases versus number of switches per layer.

switching element are present). Fig. 13 shows the results. This experiment also corresponds to the case of the largest algorithmic execution complexity as the variable N is increased (the proposed algorithm is $O(N^4WL)$. Table 3 shows the average execution time of the proposed algorithm for increasing values of N for the LINS evaluated in Fig. 13.

3. In this experiment, the variable is the number of switches in each switching element, i.e., W. The simulated LINs have two internal layers of 10 nets each, i.e., $L = 4$ and $N = 26$. Fig. 14 shows the simulation results.

The following conclusions can be drawn:

1. For the first experiment, although most of the adjacencies (over 75 percent of them) are covered by the LIN decomposition process, the number of residual adjacencies for postprocessing also increases very rapidly. The number of phases increases at a much lower rate than the number of adjacencies, thus the required number of phases is much less than the upper bound.

TABLE 3
Algorithm's Execution Time

N	Execution Time (seconds)
8	254
10	439
12	786
16	949
20	1303

2. For the second experiment, the number of residual adjacencies and the number of phases increase considerably. This occurs because the maximal number of paths per NDP set is fixed and more path sets are needed to cover the adjacencies.
3. In the third experiment, the total number of adjacencies is the same for every LIN, so there is little variation for postprocessing.
4. The results for the number of phases are not monotonic. This occurs because each NDP set may cover a number of additional adjacencies. This process cannot be fully controlled in the execution of the algorithm. Therefore, a nonmonotonic situation is indeed possible.

9 Conclusion

This paper has presented an algorithmic approach for testing layered interconnection networks (LINs). An LIN is a nonhomogeneous multistage interconnection network; it consists of nets and switching elements. Nets are arranged in layers (of different cardinality); each switching element has the same number of switches, but it may have different patterns.

A divide-and-conquer approach is proposed under a comprehensive fault model for both nets and switches (for example, bridge faults are defined at a structural level using an adjacency graph). This approach is based on the multiple execution of the maximum flow algorithm; in particular, it has been proven that, for an LIN graph G with maximal degree D_p, it is possible to find a flow to cover all nodes with degree D_p such that it is always possible to decompose G into D_p node-disjoint path (NDP) sets to cover all nodes (with reuse) and edges (without reuse) of the graph. By

construction of various graphs, it is proven that this decomposition can be achieved optimally.

The proposed approach employs a divide-and-conquer procedure made of two parts (generation of LIN configurations by graph decomposition and test vector generation) and is based on the equivalence between the LIN testing problem and the node-disjoint path (NDP) sets problem. The minimal number of NDP sets corresponds to the minimal number of configurations of the LIN; in each phase, an LIN configuration is generated by turning ON and OFF the switches and establishing the desired connections. Minimality is related to the successive decrease of the maximal degree of the graph by establishing a feasible flow and the appropriate circulation. This is achieved by constructing an appropriate graph from G and using the maximum flow algorithm with upper and lower capacity bounds. Once the configurations of G are found, then a test vector generator is used for the LIN, i.e., testing of the LIN consists of testing each configuration at a time. In each phase, the LIN is configured using an NDP set. Faults in the switches and nets are detected. To guarantee 100 percent coverage of the shorts, the maximum flow algorithm is used for processing the residual adjacency graph. By setting the lower capacity of the net with a remaining adjacency to 1, it is then possible to find a feasible flow to cover as many nets with remaining adjacencies. The complexity for decomposing the LIN is $O(\max(M, D_p)N^2WL)$, where N is the total number of nets, W is the number of switches per switching element, M is the number of executions of the postprocessing algorithm, and L is the number of layers. As test vector generation is $O(N^2)$ using [5], then the complexity of the proposed approach is $O(N^4WL)$.

Several experiments have been performed to evaluate this algorithm by simulation; in general, although most of the adjacencies (over 75 percent of them) are covered by the LIN decomposition process, the number of adjacencies for postprocessing can increase rapidly and it has been shown that the required number of phases is much less than the upper bound (as proven in this paper).

APPENDIX A

```
Network Decomposition Algorithm:
Decompose(G)
{
  Construct the graph G' from G
  Using the original adjacency graph as the
  initial residual adjacency graph
  Initialize G'
  Check current_maximal_degree
  while (current_maximal_degree != 0)
  {
    Set the lower capacity bound 1 for
    pseudoedges with maximal degree to 1
    Set all c' = c - 1
    Calculate the feasible flow using the
    Preflow-Push algorithm
    Record edges on the path of the feasible
    flow
    Update G' by removing all edges traversed by
    the feasible flow
    Update residual adjacency graph
    Generate the adjacency graph for the
    current configuration.
    Decrease current_maximal_degree by 1
    Initialize the updated G' for next
    iteration use
  }
}
```

APPENDIX B

```
Residual Adjacency Graph Processing Algorithm:
ResiGadProc(G)
{
  Construct G' from G
  Initialize G'
  Check the residual adjacency graph traversed
  by the decomposition process
  while (an adjacency exists in the residual
  adjacency graph)
  {
    Set the lower capacity bound 1 for the
    pseudoedges with untested
      adjacencies to 1
    Set all c' = c - 1
    Calculate the feasible flow using the
    Preflow-Push algorithm
    Record the edges on the path of the feasible
    flow
    Update the residual adjacency graph
    Initialize updated G' for next iteration
    use
  }
}
```

ACKNOWLEDGMENTS

This research supported in part by the ITC endowment.

REFERENCES

[1] G. Lemieux, P. Leventis, and D. Lewis, "Generating Highly Routable Sparse Crossbars for PLDs," *Proc. ACM Int'l Symp. FPGAs,* pp. 155-164, 2000.

[2] V. Betz and J. Rose, "Automatic Generation of FPGA Routing Architectures from High-Level Descriptions," *Proc. ACM Int'l Symp. FPGAs,* pp. 175-186, 2000.

[3] Y.-W. Chang, D.F. Wong, and C.K. Wong, "Universal Switch-Module Design for Symmetric-Array-Based FPGAs," *Proc. ACM Int'l Symp. FPGAs,* pp. 80-86, 1996.

[4] M.A. Khalid and J. Rose, "A Hybrid Complete Graph Partial Crossbar Routing Architecture for Multi FPGA Systems," *Proc. ACM Int'l Symp. FPGAs,* pp. 45-54, 1998.

[5] X.T. Chen and F. Lombardi, "A Coloring Approach to the Structural Diagnosis of Interconnects," *Proc. IEEE Int'l Conf. Computer-Aided Design,* pp. 676-680, 1996.

[6] C. Wu and T. Feng, "Fault Diagnosis for a Class of Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. 30, no. 10, pp. 743-758, 1981.

[7] F. Lombardi and W.-K. Huang, "On the Constant Diagnosability of Baseline Interconnection Networks," *IEEE Trans. Computers,* vol. 39, no. 12, pp. 1485-1488, 1990.

[8] C. Feng, F. Lombardi, and W.-K. Huang, "Detection and Location of Multiple Faults in Baseline Interconnection Networks," *IEEE Trans. Computers,* vol. 41, no. 11, pp. 1340-1344, 1992.
[9] W.T. Cheng, J.L. Lewandowski, and E. Wu, "Diagnosis for Wiring Interconnects," *Proc. Int'l Test Conf.,* pp. 565-571, 1990.
[10] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction To Algorithms,* pp. 579-629. New York: McGraw-Hill, 1990.
[11] A.J. Hoffman, "Some Recent Applications of the Theory of Linear Inequalities to Extremal Combinational Analysis," *Proc. Symp. Applied Math.,* vol. 10, pp. 113-127, 1960.
[12] W.H. Kautz, "Testing for Fault in Wiring Networks," *IEEE Trans. Computers,* vol. 23, no. 4, pp. 358-363, 1974.
[13] J.V. Leeuwen, *Handbook of Theoretical Computer Science,* vol. A, pp. 591-592. Cambridge, Mass.: MIT Press, 1990.
[14] F. Lombardi, X.T. Chen, and W.K. Huang, "On the Diagnosis of Programmable Interconnects: Theory and Applications," *Proc. IEEE VLSI Test Symp.,* pp. 204-209, 1996.
[15] J. Rose and S. Brown, "Flexibility of Interconnection Structure for Field Programmable Gate Arrays," *IEEE J. Solid-State Circuits,* vol. 26, no. 13, pp. 277-282, 1991.
[16] Xilinx, Inc., *The Programmable Logic Data Book.* San Jose, Calif., 1994.
[17] C.W. Yau and N. Jarwala, "A Unified Theory for Designing Optimal Test Generation and Diagnosis Algorithms for Board Interconnects" *Proc. Int'l Test Conf.,* pp. 71-77, 1989.
[18] A. Hassan, J. Rajski, and V.K. Agrawal, "Testing and Diagnosis of Interconnects Using Boundary-Scan," *Proc. IEEE Int'l Test Conf.,* pp. 126-137, 1985.
[19] C. Wu and T. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. 29, no. 8, pp. 694-702, 1980.
[20] M.A. Franklin, "VLSI Performance Comparison of Banyan and Crossbar Communication Networks," *IEEE Trans. Computers,* vol. 30, no. 4, pp. 283-290, 1981.
[21] M. Patyra and W. Maly, "Circuit Design for a Large Area High Performance Crossbar Switch," *Proc. IEEE Int'l Workshop DFT on VLSI Systems,* pp. 32-45, 1991.
[22] A. Mourad, B. Ozden, and M. Malek, "Comprehensive Testing of Multistage Interconnection Networks," *IEEE Trans. Computers,* vol. 40, no. 8, pp. 935-951, 1991.
[23] J.D. Ullman, *Computational Aspects of VLSI.* Rockville, Md.: Computer Science Press, 1984.
[24] Y. Chen, D. Lawrie, D. Padua, and P. Yew, "Interconnection Networks Using Shuffles," *Computer,* vol. 14, no. 12, pp. 55-65, 1981.

Bin Liu received the BSEE and MSEE degrees in July 1992 and July 1995, respectively, from Fudan University, Shanghai, China. From August 1995 to December 1996, he was a PhD student in electrical engineering at Auburn University, Auburn, Alabama. He was a graduate student at Texas A&M University, where he received the MS degree in computer science. From May 1997 to August 1997, he was a student IC design engineer at MOSAID Technologies, Inc., Kanata, Ontario, Canada. He has been a student member of the IEEE since 1996. His research interests include IC design and testing, micro-lithography, computer architecture, and digital system design.

Fabrizio Lombardi graduated in 1977 from the University of Essex (United Kingdom) with the BSc degree (hons.) in electronic engineering. In 1977, he joined the Microwave Research Unit at University College London, where he received the master's degree in microwaves and modern optics (1978), the diploma in microwave engineering (1978), and the PhD degree from the University of London (1982). He is currently the chairperson of the Department of Electrical and Computer Engineering and holder of the International Test Conference (ITC) Endowed Professorship at Northeastern University, Boston. Prior to that, he was a faculty member at Texas Tech University, the University of Colorado-Boulder, and Texas A&M University. He has received many professional awards: the Visiting Fellowship at the British Columbia Advanced System Institute, University of Victoria, Canada (1988), twice the Texas Experimental Engineering Station Research Fellowship (1991-1992, 1997-1998), the Halliburton Professorship (1995), and an International Research Award from the Ministry of Science and Education of Japan (1993-1999). Dr. Lombardi was the recipient of the 1985/86 Research Initiation Award from the IEEE/Engineering Foundation and a Silver Quill Award from Motorola-Austin (1996). He was an associate editor (1996-2000) of the *IEEE Transactions on Computers* and a distinguished visitor of the IEEE Computer Society (1990-1993). Since 2000, he has been the associate editor-in-chief of the *IEEE Transactions on Computers*. Currently, he is also an associate editor of *IEEE Design and Test* and a distinguished visitor of the IEEE Computer Society. He has been involved in organizing many international symposia, conferences, and workshops sponsored by professional organizations as well as guest editor of special issues in archival journals and magazines such as the *IEEE Transactions on Computers*, *IEEE Transactions on Instrumentation and Measurement*, *IEEE Micro*, and *IEEE Design and Test*. He is the founding general chair of the IEEE Symposium on Network Computing and Applications. His research interests are testing and design of digital systems, ATE systems, configurable/network computing, defect tolerance, and CAD VLSI. He has extensively published in these areas and edited six books. He is a senior member of the IEEE.

Nohpill Park received the BS degree in 1987, the MS degree in computer science in 1989, both from Seoul National University, Seoul, Korea. He received the PhD degree in 1997 from the Department of Computer Science of Texas A&M University. He is currently an assistant professor in the Computer Science Department at Oklahoma State University. His research interests include computer architecture, defect and fault-tolerant systems, testing and quality assurance of digital systems, parallel and distributed computer systems, multichip module systems, and programmable digital systems, reliable digital instrumentation. He is a member of the IEEE.

Minsu Choi received the BS, MS, and PhD degrees in computer science from Oklahoma State University in 1995, 1998, and 2002, respectively. He is currently with the Department of Electrical and Computer Engineering at the University of Missouri-Rolla as an assistant professor. His research mainly focuses on computer architecture and VLSI, embedded systems, fault tolerance, testing, quality assurance, reliability modeling and analysis, configurable computing, parallel and distributed systems, dependable instrumentation and measurement, and autonomic computing. He was the recipient of the Don and Sheley Fisher Scholarship, 2000, and Korean Consulate Honor Scholarship, 2001, and Graduate Research Excellence Award, 2002. He is a member of the IEEE, Sigma Xi, and Golden Key National Honor Society.