An Efficient Routing Method for Pseudo-Exhaustive Built-in Self-Testing of High-Speed Interconnects

J. Liu and W. B. Jone
Department of Electrical & Computer Engineering
liujn@email.uc.edu, wjone@ececs.uc.edu

Abstract

This paper presents a powerful routing method for pseudo-exhaustive built-in self-testing of high-speed interconnects with both capacitive and inductive crosstalk effects. Based on the concepts of test cone and cut-off locality, the routing method can generate an interconnect structure such that all nets can be tested by pseudo-exhaustive patterns. The test pattern generation method is simple and efficient. Experimental results obtained by simulating a set of MCNC benchmarks demonstrate the feasibility of the proposed pseudo-exhaustive test approach and the efficiency of the proposed routing method.

1. Introduction

Noise effects can cause crosstalks and signal overshoot and ringing. If the signal loss on an interconnect is out of the defined safe margin, it may cause performance degradation, even logic error [1]. Several design techniques, including physical design and analysis tools, have been developed to help design for margin and minimize crosstalk problems [2] [3]. However, it is hard to anticipate in advance the impact of a full range of all possible process variations and manufacturing defects. Due to the complexity of the signal integrity problem, it is very hard to fix it in the design phase. Hence, there is a critical need to develop testing techniques for manufacturing defects that may produce crosstalk effects.

In dealing with the signal integrity testing problem for RC interconnects, one of the most famous fault model is maximum aggressor (MA) model [4]. The basic idea of the MA fault model is to apply identical transitions to all wires except the victim line to create the maximal integrity loss in the victim line. It has been used extensively in crosstalk signal integrity testing methods [5][6]. However, it fails to deal with long range, complex inductive coupling which is significant in current Giga Hertz designs. In [7], it has been found that there exist test patterns creating worse delay and/or noise and causing more integrity loss compared to those generated by the MA model. Due to the complexity of inductive coupling in RLCK interconnects, finding test patterns guaranteed to create the worst-case scenarios for integrity loss is almost impractical. It is even concluded in [8] that random test patterns are more qualified than those based on conjectured models to create the worst-case integrity test.

There are few test pattern generation methods targeting RLCK interconnects with full consideration of long range inductive coupling effects. One interesting attempt is to use an efficient simulation method to do test pattern generation [9]. To enhance the performance, model order reduction is applied to alleviate the computation complexity with slight loss of accuracy. Due to the complexity of a real circuit, it might be hard to apply this method to a large interconnect structure in real circuit design.

The concept of pseudo-exhaustive built-in self-testing (PE-BIST) for crosstalk noises of high-speed interconnects has been proposed in [10]. PE-BIST is a natural choice for interconnect noise testing due to the local property of capacitive and inductive noises. In this paper, we mainly focus on presenting an efficient routing method based on the concepts of *pseudo-exhaustive test cone* and *cut-off locality* to form an interconnect structure which is pseudo-exhaustive testable. The rest of this paper is organized as follows. The basic ideas of pseudo-exhaustive testing and crosstalks are discussed in Section 2. Section 3 deals with PE-BIST test cone determination. Section 4 focuses on test pattern generation and delivery. Section 5 is the core part of this paper which deals with the post global routing problem to generate PE-BIST testable interconnects. Experimental results are shown in Section 6. Finally, concluding remarks are given in Section 7.

2. Background

Pseudo-exhaustive testing has many of the benefits of exhaustive testing, but the number of test patterns can be greatly reduced by applying exhaustive test patterns to each output cone, instead of the entire circuit [11][12][13]. Fortunately, for signal integrity testing, it has been observed that not all signal lines around a victim interconnect are effective aggressors for the victim under testing.

1-4244-1258-7/07/$25.00 ©2007 IEEE

In low and mid-range frequencies, capacitive coupling has been the major noise source for signal integrity problems. It is well known that capacitive coupling has local effects. The capacitive coupling effect between interconnects decreases substantially for non-adjacent lines. In a typical RC interconnect bus simulation, results show that we can test the victim line by exhaustively exercising only several nearest aggressors instead of all aggressors, without loss of the desired test accuracy. For RC-like interconnects, capacitive coupling decreases greatly with distance.

While at high frequencies, inductive coupling becomes no longer negligible, and is no longer a short range effect. In fact, the inductive coupling effect decreases slowly in space which results in a long range effect [14]. Crosstalk generally involves multiple coupled RLCK interconnects. Shielding insertion is known as an efficient technique to reduce the inductive coupling between signal wires. A shield is basically a metal directly connected to ground. A shield can reduce inductive noise because it supplies a current return path for aggressor signal wires, thus reducing coupling between signal wires. A dedicated shield is generally considered as a good current return path in high-speed interconnect design. A good shielding scheme will greatly reduce inductive coupling. It has been shown that the noise effect decreases monotonously for different aggressor groups, if shields are inserted into an interconnect structure regularly to divide aggressor wires into aggressor groups. This results in a similar local noise effect like in the RC case [10].

Naturally, we introduce the concept of *locality* as a measurement of an aggressor's significance to a victim.

Definition: Locality is the distance between an aggressor line and the victim line, expressed by the number of lines between the aggressor and the victim. For aggressors immediately adjacent to the victim line, the locality is zero.

As an aggressor is far away from the victim line, it becomes an ineffective aggressor and thus can be ignored in the test set for the victim line. Depending on the desired test accuracy, there is a milestone locality to decide which aggressors are effective ones. Beyond this locality, the aggressors can be considered insignificant for the test case. This locality is defined as *cut-off locality*.

Locality serves as a measurement of the influence on the victim line by an aggressor line in both the RC and RLCK cases. By choosing a cut-off locality, the entire exhaustive testing space can be substituted by a set of effective aggressor test spaces without loss of the desired accuracy. In [10], we have shown that for RLCK interconnects, the noise effect of aggressor groups is very similar to the noise effect of aggressor lines in RC interconnects. Due to the limited size of this paper, we cannot present the shielding insertion scheme for RLCK interconnect PE-BIST. We assume that a shielding insertion scheme already exists to divide RLCK interconnects into aggressor groups which behave similar to signal wires in RC interconnects, in terms of locality. This releases our burden of explaining RLCK shielding insertion scheme in detail. (We will cover that topic in a separate paper). We can focus on how to use the concept of *locality* to construct a pseudo-exhaustive testable interconnect structure. Readers will soon realize that our algorithm is largely dependent on the concept of *locality*. The testing method and algorithm can be applied to any kind of interconnects as long as the crosstalk effect can be confined to a finite number of *"local"* aggressors.

Based on the concept of locality, we can just apply exhaustive patterns of those effective aggressor lines to the victim line. Such a test set is called a *pseudo-exhaustive test set* of the victim line. Fig. 1 shows an example of an interconnect PE-BIST cone and the victim line is represented by "V". If each aggressor has two states (rising and falling transitions), all 2^{20} aggressor patterns are applied to both the left and right ten effective aggressor lines (each is marked by "A" in Figure 1) adjacent to victim line V. Instead of exercising full combinations of test patterns at all interconnects (except V), we apply exhaustive test patterns only to the cone of interconnects (except V) as shown in Fig. 1. This is the reason why victim line V is called pseudo-exhaustively tested. The whole test setting (one victim line and 20 aggressor lines) is called a *test cone*. Thus, a test cone of a victim line contains the victim itself and its left and right effective aggressors.

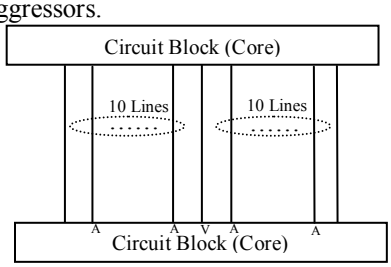

Figure 1: An interconnect PE-BIST cone.

3. PE-BIST Test Cone Determination

As we discussed in the last section, the effective aggressors of a victim line can be identified by setting its cut-off locality. Any aggressor with its locality (with respect to the victim line) less than the cut-off locality is considered as a significant aggressor, and thus should be included in the pseudo-exhaustive test cone of the victim. For a simple interconnect structure like a data bus, when the cut-off locality is set, the test cone size can be easily determined. For a simple interconnect bus structure, if the cut-off locality is set to n, the test cone size is $2n+1$ ($2n$ aggressors and one victim).

For an arbitrary interconnect structure, a long signal line may span different routing regions, and the aggressors to this line are distributed. We use Net Interference Graph (NIG) to record the crosstalk between different nets. Each node of a NIG represents an available signal net. If two nets constitute a potential aggressor-victim pair, there is an edge connecting them in the NIG. If two nets are within cut-off locality range, they are aggressor-victim pairs. Accordingly, there is an edge connecting these two nets in the corresponding NIG.

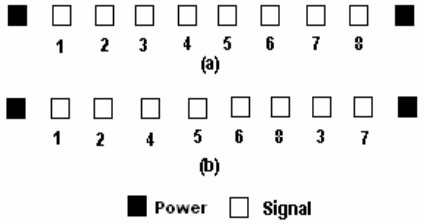

Figure 2. Example interconnect structure in two routing regions.

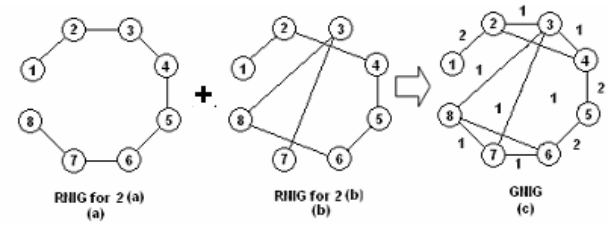

Figure 3. Merge of two regional NIGs into a global NIG.

Fig. 2(a) and Fig. 2(b) show two interconnect structures in two routing regions. Fig. 3 shows their corresponding (regional) NIGs. For simplicity, we set the cut-off locality to 1. This means we consider crosstalk effects only between adjacent nets. Fig. 3(a) shows that each node is connected to its adjacent nets in Fig. 2(a). Note that in Fig. 3(a), the NIG is constructed from a specific routing region, and is called a *regional* NIG (RNIG). So is Figure 3(b). A global NIG (GNIG) is a simple combination of all RNIGs. Fig. 3(c) shows an example of the GNIG formed from two RNIGs in Figures 3(a) and 3(b). The weight of an edge in the GNIG is the total number of times the edge appearing in all regional NIGs. In this GNIG, the weight of edges 1-2, 4-5, 5-6 is 2 (Fig. 3), since those edges appear in both RNIGs.

Once we have the GNIG formed, we can determine the effective aggressors for each victim net which in turn indicates the test cone size requirement. For example, to test net 3 in Fig 3, we need to exhaustively test its aggressor nets 2, 4, 7 and 8. The number of aggressor nets for a victim equals to the order of the victim vertex in the GNIG. To test the victim net, we must cover all its aggressors in the test set. For a node with a small order, its test set is small. The PE-BIST test cone size must be large enough to cover all the aggressors of the node which has the maximum order (called *m*). By doing so, the test set will also be able to cover all those nodes with order less than *m*. So, the *maximum order of all vertices* in the GNIG constitutes the *minimum test cone size* required by the proposed PE-BIST method.

4. PE-BIST Test Pattern Generation and Delivery

As shown in Fig. 4, the test pattern generation circuit for the interconnect PE-BIST method contains an LFSR-driven aggressor signal generator, a victim signal generator, a test pattern generation controller and a PE-BIST controller. The LFSR is used to generate all combinations of 0 and 1 logic values except the all-zero test pattern. The *aggressor signal generator* (ASG) will issue either a rising or falling transition upon excitation from the LFSR. For example, an LFSR bit with logic 0 (1) will generate a rising (falling) transition in the corresponding aggressive line. The *victim signal generator* (VSG) is used to generate signals for the victim interconnect, and the signals include Q1 (constant logic 1), Q0 (constant logic 0), rising and falling. The *PE-BIST controller* is used to coordinate the operation of the entire test pattern generation process. The output of the *test pattern generation controller* is abstracted as "Channels". The test pattern generation controller works in such a way that it will select one channel as the victim channel and connect it to the victim signal generator. All other channels are connected to the aggressor signal generator. For each victim excitation pattern, the aggressor signal generator will excite all aggressors to generate a pseudo-exhaustive test set for this victim channel. Then, it will set the next channel as the victim channel, and generate pseudo-exhaustive patterns for that channel. This process is repeated until each channel has been selected as a victim channel.

For a test cone size *n+1*, there are *n* aggressor channels and one victim channel. The total number of test patterns is $4(n+1)\times 2^n$. The reasons are: (1) there are *n+1* different victims within this cone, (2) each victim will be tested by 2^n different test patterns from *n* aggressor lines, and (3) each victim can have four different values (Q1, Q0, rising and falling) for pseudo-exhaustive test patterns from its aggressors.

Theorem 4.1 If all aggressors of a victim net and the victim net are connected to different channels of the test pattern generation controller, the victim net will be pseudo-exhaustively tested. *Proof:* omitted.

To assign nets to each channel, we begin with the node with the maximum degree of connection. Then, we assign different channel numbers to the node and all its

connected nodes. For the example interconnect structure in Fig. 5, nets 3, 4, 2, 7, 8 are assigned to C1, C2, C3, C4, C5, respectively. For nets 1 and 5, they are assigned C5 while net 6 is assigned to C3. There are multiple ways to assign nets to channels. As long as all connected nets are assigned different channels, they will be pseudo-exhaustively tested during the test pattern generation process. For example, when channel 3 is assigned victim test patterns (nodes 2 and 6 are victim nets), channels C1, C2, C4, C5 will apply exhaustive aggressor patterns to aggressor nets 1, 3, 4, 5, 7, 8. Fig 6 shows the connection of channels to the interconnect signal lines. Note that nets 2 and 6 (1, 5, and 8) are tested simultaneously.

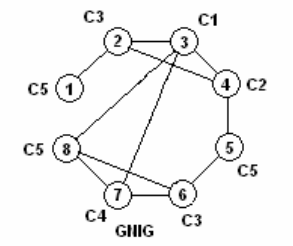

Figure 5. PE-BIST test pattern channel assignment.

The test pattern generator can be distributed to different circuit cores. There is no need for one central test pattern generator. All test pattern generators are synchronized such that any channel with the same label will generate the same test pattern at any given time. Each signal net driving an output cell from any circuit core will be connected to the corresponding channel determined by the labeling process discussed above.

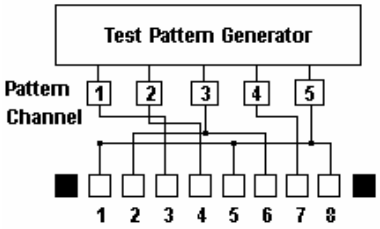

Figure 6. PE-BIST test pattern delivery.

Fig. 7 shows how the test pattern generators and response analyzers are distributed in different circuit cores. The figure shows how test patterns generated from different cores can come together to form a pseudo-exhaustive test cone. For the test cone shown in Fig. 7, the test cone is formed by signal nets from different circuit cores. Specifically, circuit 1 supplies channels C4 and C5 while circuit 2 supplies channels C1, C2 and C3. Since those channels are independent from each other in the test pattern generation process, they form a pseudo-exhaustive test set.

In order to coordinate the entire test process, there is a centralized test controller. The controller controls the entire test process: setting the victim channel, triggering the distributed test pattern generators to start, and collecting the circuit responses after pseudo-exhaustive test patterns are applied. Note that all test generators are synchronized, and generate the same test pattern in each clock cycle. For each victim channel under testing, the corresponding response analyzer will collect the channel responses by compressing them [5][7][8]. Once the victim channel is pseudo-exhaustively tested, the PE-BIST controller can then collect all responses from all distributed response analyzers and perform central fault processing. Note that a victim channel may drive several victim lines. That is the reason why several test response analyzers may be involved.

The whole testing process is straightforward. It would be easy to reuse the existing BIST circuits, e.g., the pseudo-exhaustive test pattern generators and the signature analyzers that have been built in each core. The most important thing in organizing the PE-BIST architecture is the assignment of each net into a track by interconnect routing, and the assignment of a test channel for each net. Both assignments of tracks and channels must accomplish the goal of pseudo-exhaustive testing with a specific requirement of the cut-off locality and test cone size.

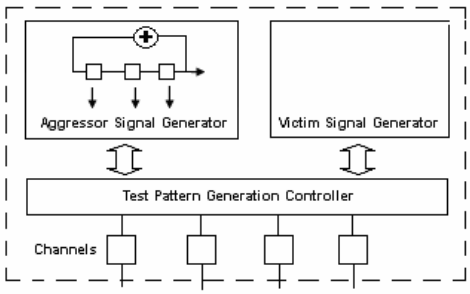

Figure 4. Logic structure for PE-BIST test pattern generation.

5. Post Global Routing for PE-BIST

Sections 3-4 discussed the basic idea of PE-BIST for interconnect testing. Especially, Section 3 discussed how to determine the PE-BIST test cone size, and in reality the test cone size cannot be too large. Constrained by test time, the total number of pseudo-exhaustive test patterns for each test cone cannot exceed a specific number, e.g. 2^{30}. If we have a GNIG with the maximum degree of all vertices larger than 30, we have to find a way to control the degree of that vertex so that it meets the test time constraint.

In modern routing technologies, the routing problem is usually solved by using a two-stage approach: global routing followed by detailed routing. The detailed routing step includes track assignment and detailed net routing. The relative physical track of each net plays a crucial role

in determining the NIG, which subsequently determines the PE-BIST test cone. In order to make an interconnect structure PE-BIST testable with a reasonable test cone size, we must have control over the routing phase, especially track assignment. So, we detach the track assignment process from the detailed net routing phase as an independent and intermediate step between global routing and detailed routing process. This section will discuss how to finish tracking assignment in a way that a PE-BIST solution with a limited test cone size is possible.

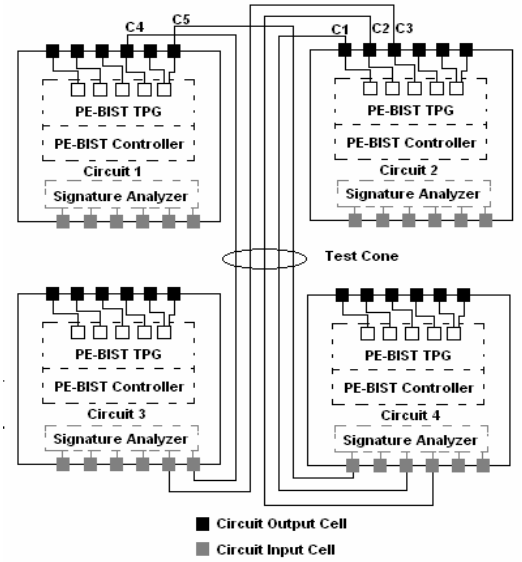

Figure 7. PE-BIST distributed test pattern generation.

5.1 PE-BIST Routing Problem

Formulation of the PE-BIST Routing Problem: Given a set of signal nets, their global routing solution, and a specific pseudo-exhaustive test cone size with the desired cut-off locality, the PE-BIST routing problem is to find a net-track assignment solution so that all nets are PE-BIST testable within the test cone size.

After global routing, each net is assigned to the individual routing region but not yet put on a track. The track assignment step will determine the NIG (both RNIG and GNIG). If we want to have nets to be aggressor-victim pairs, we can place them close to each other within the cut-off locality range. On the other hand, if we want to decouple two nets, we can put them far away from each other. This will alter the topology of the NIG, which subsequently decides the PE-BIST test cone size. By placing nets into the right tracks, we can have control over the NIG and ultimately over the PE-BIST test cone size. This is why we choose to construct PE-BIST testable interconnects in the post global routing phase.

In the process of building the GNIG, we first build the RNIG for each region individually, and then combine all RNIGs into the GNIG. In our track assignment algorithm, we follow the same bottom-up building process: starting with RNIG construction and then accumulating RNIGs into the GNIG gradually. First, we start with a routing region which has the least number of nets. We choose such a routing region to start because it has the least flexibility in placing those nets in the region. We place nets into tracks one by one for each region. The net is chosen by a cost function such that the net with the least cost and satisfying the GNIG PE-BIST constraint will be selected. The cost calculating process is repeated for every net placement to dynamically reflect the cost change after a net is placed. If no candidate net can be found for a certain track because of the GNIG PE-BIST constraint, a shield can be inserted to decouple certain nets placed in the region. In the next section, we will illustrate the algorithm with an example.

5.2 PE-BIST Routing Algorithm

Assume we have a GNIG processed so far given in Fig 8. That is, currently, seven nets 1 to 7 have been assigned to tracks in one or more regions, and their corresponding relationship is presented in Fig. 8. The test cone size is preset to 5 with the cut-off locality set to 2. Now, we need to process a region which contains nets 4, 5, 6, 7, 8 after the global routing phase. Our mission is to find a track assignment and shield insertion solution so that the new GNIG still satisfies the PE-BIST requirement, i.e., the max order of each vertex is less than the pseudo-exhaustive test cone size: 5.

First, we extract the available NIG information for the nets in the processed region from the GNIG. It is essentially a sub-graph of the GNIG except that we add a new vertex if the corresponding net was not present in the GNIG. Fig. 9 shows the NIG information extracted from the GNIG with a new node 8.

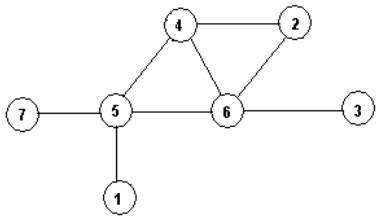

Figure 8. An example GNIG as starting point

Our next step is to place nets (4, 5, 6, 7, 8) into different tracks so that the resultant GNIG still satisfies the PE-BIST requirements. If we can find a net order such that the GNIG remains the same, we say that it is *cost-free* to incorporate this region into the GNIG. Most of the time, we need to either modify the GNIG by adding new edges (newly formed aggressor-victim relationships) or we need to add extra shields to prevent nets from interfering with each other. To illustrate this point, we use the above example again.

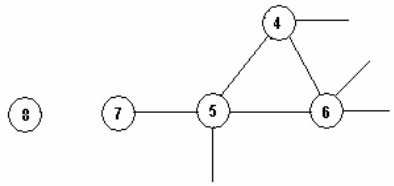

Figure 9. PE-BIST Routing example GNIG (cont.).

First, we need to find the max clique in the extracted NIG. A clique of a graph G requires that every node in the clique (which is a sub-graph of G) is connected to each other. In the NIG, this means every node (representing each individual net) is allowed to interfere with each other. Since we use a bottom-up approach, the clique information can be updated every time when we add a new edge to the graph, without exhaustively searching for cliques. We have a clique in Fig. 9 formed by nets 4, 5, 6. If several cliques are found with the same number of nodes, we select the one whose sum of orders is maximum. If a clique is not found, we select a two-node pair whose sum of orders is maximum. In our example, we have only one clique which contains nets 4, 5, 6. In this clique, we further sort the vertices by their orders.

It is required that the order of every vertex in the GNIG be less than the test cone size. On the other hand, if a vertex's order is less than the test cone size, the net represented by that vertex can always be pseudo-exhaustively tested. While we must limit the maximum order of each vertex, we can add more edges to a vertex as long as its order is smaller than the test cone size. If a vertex has the maximum order, the corresponding net is placed first since it has fewer freely available connections. If two nets have the same order, we place the net with less connections to nets in this region first. In our example, both nets 5 and 6 have the highest order, followed by net 4. But net 6 has two connections to nets (4, 5) in this region, while net 5 has three connections to nets (4, 6, 7) in this region. So, net 6 is placed first, followed by net 5. The reasoning behind this is that nets with less connections to nets in the region have a greater probability of introducing new edges in the NIG when interacting with the rest of the nets in the region. It is placed first to reduce its exposure to those nets.

Now, we have an initial track assignment for nets 6-5-4 adjacently. Since we set the cut off locality to 2, if we add another net adjacent to net 4, that net will interfere with both nets 5 and 4. If adding such interference causes the order of a vertex greater than or equal to the pseudo-exhaustive test cone size, it is not allowed to be added and shield(s) must be added to avoid that interference. On the other hand, if it does not result in that way, the net can be placed adjacent to net 4. To identify which net is the best candidate to be placed aside net 4, we first sort all available nets according to the cost of placing each of them adjacent to net 4.

The cost considers two factors: (1) the number of edges to be added into the NIG, and (2) the number of shields to be added to separate the nets. In our example, to add net 7 aside net 4, we have to add a new edge between nets 7 and 4. While to add net 8 aside net 4, we need to add two edges: one is between nets 8 and 4, while the other is between nets 8 and 5. Net 5 will have an order of 5 which is equal to the test cone size. So in order to put net 8 aside net 4, a shield must be added between net 8 and 4. The cost of adding a shield is more than the cost of increasing number of edges in the NIG. Net 7 has the minimum cost. So we place net 7 adjacent to net 4 which results in Fig 10. Note that we added one new edge between nets 7 and 4. Our track assignment becomes nets 6-5-4-7.

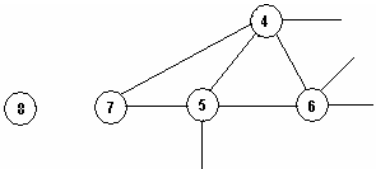

Figure 10. PE-BIST Routing example GNIG (cont.).

Now, only net 8 is left. We have to place net 8 aside net 7. But if we place net 8 aside net 7, net 8 will have interference with both nets 7 and 4 which results in adding two new edges in the NIG. It will force the vertex representing net 4 to have an order of 5 which is no smaller than the test cone size 5. This is prohibited. So, we have to place a shield between nets 7 and 8. By placing a shield between nets 7 and 8, there will be no added edges in the NIG since net 8 will not interfere with either net 7 or 4. The resulting track assignment is thus nets 6-5-4-7-8, and the resultant GNIG is shown in Fig 11. Comparing Fig. 8 and Fig. 11, one can find out that only one new edge and one new shield are added in the GNIG after the track assignment in this region. By repeating the above algorithm to all regions, we can guarantee to get a GNIG with the maximum order of all vertices less than the test cone size.

The last step is channel assignment. As we have explained in Section 4, for a given test cone size *n*, our PE-BIST test generator must have *n* channels. For any net in the GNIG, as long as the net and all its aggressors are assigned different channels, all effective aggressors are guaranteed to be involved in generating the pseudo-exhaustive test patterns for the net. For the above example, the test cone size is 5 so there are five channels C1, C2, C3, C4, C5. We start with a vertex with the maximum order, net 5 for example, and assign it to channel C1. Then, its adjacent nets 1, 4, 6, 7 can be assigned to C2-C5 (Fig. 11). We have nets 4, 6 assigned to C3 and C4, so net 2 must be assigned to either C1, C2 or C5. We randomly choose C1. We keep assigning channels until all nets are assigned. The channel assignment problem can be solved by graph-coloring methods [15]. In summary, two nets

can be assigned the same channel, if they are not connected in the GNIG. The final channel assignment is shown in Fig. 11, while the algorithm is shown in Fig. 12. We emphasize that the victim line with the largest test cone size dominates the test application time. Thus, serial test application for different regions (with different test cone size distribution) cannot reduce the test time.

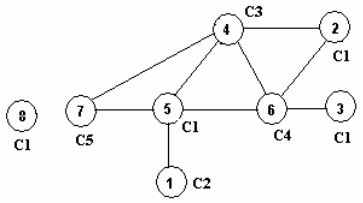

Figure 11. PE-BIST Routing example GNIG (cont.).

Algorithm: PE-BIST Routing Algorithm

Input: 1. Routing region information: R;
2. Pseudo-exhaustive test cone size : n;

Output: Track assignment for each net and possible shield insertion in each region;

Variable:
gNIG : global NIG;
rNIG : regional NIG;
track : net track assignment for R;
netProcessList : list of nets to be processed;

Sort routing regions in R by the number of nets placed in each routing region;

While (R not empty)
Select routing region Ri with the least number of Nets;
Insert all nets of Ri to netProcessList;
Extract rNIG from gNIG for region Ri;
Find the max clique in rNIG;
Sort vertices in the clique by the order of each vertex, and the number of its connections to vertices in current rNIG (if necessary);
Assign vertices in the clique to tracks;
Delete all nets in the clique from netProcessList;
While (netProcessList not empty)
For (each vertex in netProcessList)
Placement cost = number of edges and shields added if placed in the track under consideration;
Select vertex Vj with the least cost;
If (order of any vertex in gNIG >= cone size n)
add one shield into track;
Assign vertex Vj into track;
Delete vertex Vj from netProcessList;
Update rNIG and gNIG to reflect the added edges and shields;
End For
End While
End While
Assign channels to each net with the final gNIG;
End Algorithm

Figure 12. PE-BIST routing algorithm

6. Experimental Results

The routing algorithm for PE-BIST testable interconnects has been implemented in C++ and tested on a P4, 2.6 GHz, 1G memory, IBM PC. Since we could not find routing benchmarks for SoC circuits, we selected several MCNC benchmarks which contain a number of macro cells to simulate SoC circuits. The specification of each circuit is listed in Table 1. The grid size refers to the size of a global routing graph which is specified by the numbers of rows and columns. Each MCNC benchmark is placed by an SA-based floorplanner [16], and then gridized to form a global routing graph. Since the power lines have been pre-routed in our *Over The Cell* (OTC) model, we skip the routing of possible power lines in the net list. The global routing is completed by a maze router [17].

Table 2 shows how cut-off locality affects the number of shields required to route the PE-BIST testable interconnects for benchmark Xerox. Here, overhead is defined as the total shield length divided by the total signal net length. It can be observed that the overhead increases steadily as the cut-off locality increases. This is due to the fact that a net is exposed to more aggressors as the locality increases. This creates more edges in the NIG under construction, and thus more shields are used to limit the order of each vertex in the NIG. We emphasize again that the cut-off locality determines whether two nets are aggressor-victim pairs. The larger the cut-off locality is, the more nets are potential aggressor-victim pairs. This results in more connections between nodes in the NIG. However, the test cone size determines the maximum allowable order of each node in the NIG. More potential aggressor-victim pairs lead to more shields inserted in order to control the maximum order of each node within the test cone size.

Test cone size also has effects on the shielding overhead. As the test cone size decreases, the maximum allowable aggressors (i.e., the vertex degree) are limited, and thus more shielding insertions are required to control the maximum order of vertices in the NIG. Table 3 shows the shield overhead with the test cone size from 10 to 30 and the cut-off locality equal to 2. The corresponding shielding overhead decreases from 19.6% to 2.0%.

Other MCNC benchmark circuit results are listed in Table 4 and Table 5 respectively. Table 4 shows the result with the cut-off locality equal to 1, while Table 5 with the cut-off locality equal to 2. As we can see, with the test cone size equal to 30, the shielding overhead in each case is very small, less than 7%. For ami33 and apte, no shielding overhead is induced. Note that the cut-off locality with a value 1 or 2 is reasonable for RC interconnects as discussed in Section 2. It is also reasonable for RLC interconnects based on the concept of aggressor group (Section 2).

Table 1. List of MCNC benchmark circuits

Chip	# Macro Cell	# Nets	# Pins	Grid Size
ami33	33	119	442	10x8
ami49	49	408	953	38x39
apte	9	94	266	52x52
hp	11	83	309	25x21
xerox	10	195	696	29x32

Table 2. Shielding overhead vs cut-off locality

Xerox, test cone size=30, net length=4219		
Cut-Off Locality	Shield Length	Overhead
1	15	0.4%
2	84	2.0%
3	152	3.6%
4	265	6.3%
5	271	6.4%
6	356	8.4%

Table 3. Shielding overhead vs test cone size

Xerox, cut-off locality=2, net length=4219		
Test Cone Size	Shield Length	Overhead
30	84	2.0%
24	164	3.9%
20	253	6.0%
14	526	12.5%
10	830	19.6%

Table 4. MCNC benchmark results (with cut-off locality=1, test cone size=30)

Chip	Net Length	Shield Length	Overhead
ami33	638	0	0%
ami49	9354	226	2.4%
Apte	3876	0	0%
Hp	1522	0	0%
Xerox	4219	15	0.4%

Table 5. MCNC benchmark results (with cut-off locality=2, test cone size=30)

Chip	Net Length	Shield Length	Overhead
ami33	638	0	0%
ami49	9354	576	6.2%
apte	3876	0	0%
hp	1522	3	0.2%
xerox	4219	84	2.0%

7. Conclusion

In the paper, we proposed an efficient routing method for PE-BIST to test high speed interconnects. The PE-BIST concept and the related routing method can be applied to any interconnect structure which exhibits crosstalk locality. It can be applied to RC interconnects as well as RLCK interconnects with proper shielding insertion. With the aid of the global NIG and regional NIGs, the routing algorithm can efficiently assign interconnects to different tracks such that the number of shields added is as small as possible, while the goal of pseudo-exhaustive testing for interconnects can be achieved with reasonable test application time. Simulation results demonstrate the feasibility of the proposed interconnect test approach and the routing method.

Acknowledgements

This work was funded in part by National Science Foundation, USA, under grant CCF-0541103.

References

[1] L. Green, "Simulation, modeling and understanding the importance of signal integrity," *IEEE Circuit and Devices Magazine*, vol. 15, issue 6, pp. 7-10, Nov. 1999.

[2] J. Xiong, L. He, "Full-chip routing optimization with RLC crosstalk budgeting," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 366-377, March 2004

[3] N. Hanchate, N. Ranganathan, "Simutaneous interconnect delay and crosstalk noise optimization through gate sizing using game theory," *IEEE Trans. on Computers*, vol. 55, no. 8, pp. 1011-1023, August 2006.

[4] M.. Cuviello, S. Dey, X. Bai, and Y. Zhao, "Fault modeling and simulation for crosstalk in system on chip interconnects," In *Proc. International Conf. on Computer-Aided Design*, pp. 297-303, 1999.

[5] X. Bai, S. Dey and J. Rajski, "Self-test methodology for at-speed test of crosstalk in chip interconnects," in *Proc. Design Automation Conf.,* pp. 619-624, 2000.

[6] L. Chen, X. Bai and S. Dey, "Testing for interconnect crosstalk defects using on-chip embedded processor cores," in *Proc. Design Automation Conf.,* pp. 317-322, 2001.

[7] M. Nourani and A. Attarha, "Built-in self-test for signal integrity.", In *Proc. Design Automation Conf.*, pp. 792-797, 2001.

[8] A. Attarha and M. Nourani, "Testing interconnects for noise and skew in giga Hertz SoCs.", In *Proc. International Test Conf.*, pp. 305-314, 2001.

[9] A. Attarha and M. Nourani, "Test pattern generation for signal integrity faults on long interconnects," In *Proc. VLSI Test Symposium*, pp. 336-341, 2002.

[10] J. Liu, W. B. Jone, and S. R. Das, "Pseudoexhaustive built-in self-testing of signal integrity for high-speed SoC interconnects," *Instrumentation and Measurement Technology Conference,* May 2007.

[11] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital systems testing and testable design*, Computer Science Press, New York, NY, 1990.

[12] E. J. McCluskey, "Verification testing - a pseudoexhaustive test technique," *IEEE Trans. on Computers*, vol. C-33, no. 6, pp. 541-546, June 1984.

[13] E.J. McCluskey and S. Bozorgui-Nesbat, "Design for autonomous test," *IEEE Trans. on Computers*, vol. C-30, no. 11, pp. 866-875, Nov. 1981.

[14] H. Lei and L. M. Kevin, "Simultaneous shield insertion and net ordering for capacitive and inductive coupling minimization," *ACM International Symposium on Physical Design*, 2000: 55-60

[15] J. A. Bondy and U. S. R. Murty, Graph theory with applications, North-Holland, New York, 1982.

[16] http://www. cse.ucsc.edu/research/surf/GSRC/progress.html

[17] C.Y. Lee, "An algorithm for path connection and its application," *IRE trans. on Electronic Computer*, vol. EC-10, pp. 346-365, September 1961