AUTONOMOUS - TOOL FOR HARDWARE PARTITIONING
IN A BUILT-IN SELF-TEST ENVIRONMENT

Chien-In Henry Chen, Joel Yuen and Ji-Der Lee
Department of Electrical Engineering
Wright State University
Dayton, OH 45435

Abstract

A network partitioning tool, named Autonomous, is presented in this paper to partition a digital combinational portions of the circuit into different structural subcircuits so that each subcircuit can be pseudo-exhaustively tested. A Built-In Self-Test (BIST) design generator, named BISTSYN, has been developed and implemented to facilitate the BIST design with this partitioning methodology. When used with BISTSYN on a set of benchmark examples, the experimental results show that Autonomous is feasible for very large designs and the number of undetected faults are significantly reduced after the circuit is partitioned.

1. Introduction

BIST has been proposed as a powerful solution to VLSI testing problem and pseudo-exhaustive test is a BIST design methodology that provides effective, 100-percent fault coverage for all testable stuck-at faults without the need for fault simulation or deterministic test generation. A design generator, named BISTSYN, based on Three-Phase Cluster Partitioning algorithm, was developed [1] to automatically design a pseudo-exhaustive test generator for BIST. The experimental results show that BISTSYN minimizes the number of test signals that are required for pseudo-exhaustively testing a circuit effectively and produces test generation circuitry with low hardware overhead. The basic test scheme is shown in Figure 1(a) where we use i test signals instead of a and the examples of sharing test signals and linear sum are shown in Figures 1(b) and 1(c). However, for those conventional circuits which are extremely unsuitable for pseudo-exhaustive testing, i.e., there exists at least one output of such a circuit that is functionally dependent on all the inputs, Autonomous is presented to employ a hardware partitioning scheme such that the combinational portions of the circuit are partitioned into different structural subcircuits and therefore, each of which can be pseudo-exhaustively tested. In fact, it is necessary to have each partitioned subcircuit with the property that each output is functionally dependent on sufficiently small number of inputs. Access to the embedded inputs and outputs of each subcircuit can be achieved by inserting multiplexers (MUXs) and connecting the embedded inputs and outputs of each subcircuit to those primary inputs and outputs that are not used by the subcircuits.

2. Autonomous: Network Partitioning

The pseudo-exhaustive testing is not suitable to "total dependency" circuits in which at least one output is functionally dependent on all the inputs. Even for a "partial dependency" circuit, the size of a pseudo-exhaustive test set may still too large to be applicable in practice. In such cases, pseudo-exhaustive test can be achieved by partitioning techniques [2]. Moreover, since the time complexity of test generation and fault simulation grows faster than a linear function of circuit size, it is cost-effective to partition large circuits to reduce these costs.

The main problem with this technique is that, in general, the inputs of a subcircuit are not primary inputs (PIs) or its outputs are not primary outputs (POs). We need a partitioning technique to control the subcircuit outputs so that they are dependent on a small number of its inputs. Autonomous is a CAD tool presented to employ a network partitioning scheme such that the combinational portions of the circuit are partitioned into different structural subcircuits and therefore, each of which can be pseudo-exhaustively tested. Access to the embedded inputs and outputs of each subcircuit can be achieved by inserting multiplexers (MUXs) and connecting the embedded inputs and outputs of each subcircuit to those primary inputs and outputs that are not used by the subcircuits. The basic scheme of partitioning a circuit into two subcircuits is shown in Figure 2. In Figure 2(a), we show that a combinational logic has been partitioned into two subcircuits $C_1$ and $C_2$; many such partitions exist for large scale multiple-output circuits. Figure 2(b) depicts this network partitioning scheme, where multiplexers are inserted between two subcircuits, and $A'$ and $C$ respectively represent a subset of the signals in $A$ and $C$ used in testing $C_2$ and $C_1$ respectively. By controlling the multiplexers, all the inputs and outputs of each subcircuit can be accessed using primary input and output lines. For example, to test subcircuit $C_1$ the multiplexers can be controlled as depicted in Figure 2(c) to completely access all the inputs and outputs. The partitioning scheme enhances the controllability and observability of inputs and outputs associated with $C_2$ and $C_1$ respectively, and therefore the fault coverage is increased as shown by the simulation results in Section 3.

We use the following graph model to represent a circuit. A combinational circuit $C$ is represented by an undirected graph $G$ which has nodes $N = \{1, ..., n\}$ and edge set $E$. The nodes are corresponding to PIs or POs or gates of $C$. An undirected edge of $G$ represents a
signal flow between the corresponding components of C. For a large scale network, our interest is to partition the nodes of G into disjoint subsets S_1, S_2, ..., S_k, k ≥ 2, in the following way:

1. The maximum output dependency value w_j for each subgraph G_j, j = 1, ..., k, is less than or equal to the user specified size 1 of LFSR. In the other words, the number of reduced exhaustive test patterns of each subcircuit is not greater than 2^l.

2. The number of edges joining nodes in distinct subgraphs of the partition is minimized. In the other words, the number of MUXs added between distinct subcircuits is minimized.

In this paper, we'll explore the graph partitioning techniques [3,4] to achieve the above partitioning objectives.

We'll first discuss the partitioning method by partitioning a circuit into two subcircuits (k=2) as an example.

Let N = S_1 ∪ S_2 be a partition of the nodes of G. Let

\[ X_1 = (x_{11}, x_{12}, \ldots, x_{1n})^T \]
\[ X_2 = (x_{21}, x_{22}, \ldots, x_{2n})^T \]

be a indicator vector for S_1 and S_2, respectively. Thus,

\[ x_{ij} = \begin{cases} 
1 & \text{if } i \in S_j \\
0 & \text{if } i \notin S_j 
\end{cases} \]

\[ \sum_{i=1}^{n} x_{ij} = |S_j| = m_j \quad \text{for } j = 1, 2, \]

where m_j is the number of nodes in the j-th subcircuit. That is

\[ x_{11} + x_{21} + \cdots + x_{1n} = |S_1| = m_1 \]
\[ x_{12} + x_{22} + \cdots + x_{1n} = |S_2| = m_2 \]

and

\[ \frac{1}{2} \sum_{j=1}^{2} x_{ij} = 1 \quad \text{for } i = 1, \ldots, n. \]

Let a_{ij} denote the number of edges connecting nodes i and j and let A = (a_{ij}) denote the adjacency matrix for G. The number of edges with both endpoints in S_1 is given by

\[ \frac{1}{2} \sum_{r \in S_1} \sum_{s \in S_1} a_{rs} = \frac{1}{2} \sum_{r=1}^{n} \sum_{s=1}^{n} a_{rs} \times x_{r1} \times x_{s1} = \frac{1}{2} X_1^T A X_1 \]

The number of edges with both endpoints in S_2 is given by

\[ \frac{1}{2} \sum_{r \in S_2} \sum_{s \in S_2} a_{rs} = \frac{1}{2} \sum_{r=1}^{n} \sum_{s=1}^{n} a_{rs} \times x_{r2} \times x_{s2} = \frac{1}{2} X_2^T A X_2 \]

Thus, the number of edges not cut by the partition is denoted as E_u and calculated in the following:

\[ E_u = \frac{1}{2} X_1^T A X_1 + \frac{1}{2} X_2^T A X_2 \]

Let E_c denote the number of edges cut. We know E_c + E_u = |E|. Therefore, minimizing E_u is equivalent to maximizing E_c. The problem can be reformulated as the following.

\[ \text{Maximize} \quad \sum_{j=1}^{3} X_j^T \times A \times X_j \]

Subject to

\[ \sum_{i=1}^{n} x_{ij} = m_j \quad \text{for } j = 1, 2, \ldots, k. \]

where w_1, w_2 is maximum dependency of C_1 and C_2 respectively, and l is the user specified size of LFSR.

Based on the above approach, we can reformulate the problem of graph partitioning in general case, i.e., k > 2. Let N = S_1 ∪ S_2 ∪ ... ∪ S_k be a partition of the nodes of G. Let

\[ X_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^T \]

be a indicator vector for S_j, j = 1, ..., k. Thus,

\[ x_{ij} = \begin{cases} 
1 & \text{if } i \in S_j \\
0 & \text{if } i \notin S_j 
\end{cases} \]

\[ \sum_{i=1}^{n} x_{ij} = |S_j| = m_j \quad \text{for } j = 1, \ldots, k. \]
\[ \frac{1}{k} \sum_{j=1}^{k} x_{ij} = 1 \quad \text{for } i = 1, \ldots, n. \]

The number of edges with both endpoints in S_j is given by

\[ \frac{1}{2} \sum_{r \in S_j} \sum_{s \in S_j} a_{rs} = \frac{1}{2} \sum_{r=1}^{n} \sum_{s=1}^{n} a_{rs} \times x_{jr} \times x_{js} = \frac{1}{2} X_j^T A X_j \]

Thus, the number of edges not cut by the partition is denoted as E_u and calculated in the following:

\[ E_u = \frac{1}{2} \sum_{j=1}^{k} X_j^T \times A \times X_j \]

Let E_c denote the number of edges cut. We know E_c + E_u = |E|. The problem can be reformulated as the following.

\[ \text{Maximize} \quad \sum_{j=1}^{k} X_j^T \times A \times X_j \]

Subject to

\[ \sum_{i=1}^{n} x_{ij} = m_j \quad \text{for } j = 1, \ldots, k. \]
3. Simulation Results

Example 1: Figure 3 is an example circuit used for illustration of the network segmentation [5]. The circuit is a "total dependency" circuit and has 13 inputs and 2 outputs and a total number of 39 nodes in its corresponding graph model. Without the network partition, 13 test signals ($2^{40} = 1,099,511$ test patterns) are required for exhaustively testing the circuit. While using Autonomous, the circuit is partitioned into two subcircuits $C_1$ and $C_2$ shown in Figure 3 where $C_1$ has 11 inputs and 4 outputs and $C_2$ has 6 inputs and 2 outputs. By mapping the partitioned circuit to Figure 2(a), $A$ represents the signals to PIs $3-13$; $C$ represents the signals to PI 1 and PI 2; $D$ is the set of nodes: 64, 67, 70, 73 which are depicted in Figure 3. $G$ represents the signals from POs. The maximum dependency of $C_1$ is $w_1 = 4$ and that of $C_2$ is $w_2 = 6$. After the Three-Phase Cluster Partitioning algorithm is applied, it is found that 4 test signals ($2^4$ test patterns) are enough for exhaustively testing $C_1$. That is, during the testing, PIs 1, 2, 6, 10 and 12 share a test signal, and PIs 3, 7 and 13 share a test signal, and PIs 4, 8 and 11 share a test signal, and PIs 5 and 9 share a test signal. The maximum dependency of $C_2$ is $w_2 = 6$. After the Three-Phase Cluster Partitioning algorithm is applied, 6 test signals ($2^6$ test patterns) can exhaustively testing $C_2$. So $2^4 + 2^6 = 80$ test patterns are enough to pseudo-exhaustively test the circuit, which is much less than $1,099,511$ test patterns required by exhaustive test and is also favorably compared to 96 test patterns required by the pseudo-exhaustive technique [5].

Example 2: Figure 4 is an 8-bit adder/subtractor. The circuit is a "total dependency" circuit and has 18 inputs and 9 outputs and a total number of 121 nodes in its corresponding graph model. Without the network partition, 18 test signals ($2^{18} = 262,144$ test patterns) are required for exhaustively testing the circuit. While using Autonomous, the circuit is partitioned into two subcircuits $C_1$ and $C_2$ shown in Figure 6 where $C_1$ has 18 inputs and 25 outputs and $C_2$ has 25 inputs and 9 outputs. By mapping the partitioned circuit to Figure 2(a), $A$ represents the signals to all PIs; $B = C = E = F = G = D$; $D$ is the set of nodes: 85, 160, 211, 213, 167, 207, 270, 172, 288, 289, 180, 200, 206, 185, 224, 227, 193, 275, 279, 198, 302, 305, 306, 308, 309 which are depicted in Figure 6. $G$ represents the signals from POs. The maximum dependency of $C_1$ is $w_1 = 9$ and that of $C_2$ is $w_2 = 6$. After the Three-Phase Cluster Partitioning algorithm is applied, it is found that 9 test signals ($2^9$ test patterns) are enough for exhaustively testing $C_1$. During the testing, PIs $a_1, a_2, a_3$ and sub alone takes a test signal and PI sub alone takes a test signal and PIs $a_1, a_2, a_3$ share a test signal and PIs $b_1, b_2, b_3$ share a test signal and PIs $a_4, a_5, a_6$ share a test signal and PIs $b_4, b_5, b_6$ share a test signal and PIs $a_7, a_8, a_9$ share a test signal. The maximum dependency of $C_1$ is $w_1 = 9$. After the Three-Phase Cluster Partitioning algorithm is applied, 9 test signals ($2^9$ test patterns) can exhaustively testing $C_2$. During the testing of $C_2$, node 85 alone takes a test signal and nodes 160, 211, 267, 170, 172, 288, 290, 296, 306 share a test signal and nodes 203, 289, 227, 185, 275, 305, 309 share a test signal and nodes 167, 180, 224, 173, 279, 206 share a test signal, node 198 alone takes a test signal and node 302 alone takes a test signal. So $2^9 + 2^9 = 576$ test patterns are enough to pseudo-exhaustively test the circuit, which is much less than $2^{18} = 262,144$ test patterns required by exhaustive test.

Some benchmark experimental results obtained with our preliminary implementations are shown in Table 1 and 2. Table 1 compares the number of required test patterns for the single stuck-at faults in these benchmarks. In Table 1, the first column are results from "STG3" [6] which is a test generation system and is about one order faster than GENTEST [7]. The second and third columns are ATPG results (employing the FAN algorithm [8]) with and without Autonomous respectively. As seen from the Table 1, the number of untestable faults plus dropped faults are greatly reduced for some benchmarks after they are partitioned. "ATPG with Autonomous" has the highest fault coverage in all the circuits.

For some partitioned subcircuits of benchmarks, the size of pseudo-exhaustive test generator may be still relatively large even the reduction of the number of required test signals by Three-Phase Cluster Partitioning algorithm is applied. For these type of subcircuits, the fault simulation based on the designed pseudo-exhaustive test generator is performed. Table 2 compares the fault simulation results for these benchmarks. The fault coverage is calculated as the percentage of detected faults out of the number of "equivalent" single stuck-at faults and the fault efficiency is calculated as the percentage of detected faults out of all detectable faults. In Table 2, the first column are results from "Embedding BIST" [9]. The second and third columns are the fault simulation results with and without Autonomous by BISTSYN which are respectively designated by "BISTSYN without Autonomous" and "BISTSYN with Autonomous". Note that in the column of "BISTSYN with Autonomous", the number of generated tests is calculated by the sum of the number of test patterns for testing each subcir-
cuit and the fault coverage and the fault efficiency are calculated based on the sum of the number of equivalent faults and the sum of the detectable faults in each subcircuit, respectively. As shown in Table 2, "BISTSYN with Autonomous" achieves the highest fault coverage in all the circuits after a reasonable number of test patterns are generated from the designed pseudo-exhaustive test generator. After the circuits are partitioned, the number of test signals that are required for pseudo-exhaustively testing these circuits is significantly reduced in comparison with their primary inputs. This is indicated in the subcolumn "Max overhead" of Table 2, which is the maximum size of LFSRs used in self-testing the partitioned subcircuits. The subcolumn "Max overhead" indicates the overhead of multiplexers (calculated in terms of transistor pairs) inserted between distinct subcircuits. The average overhead of multiplexers in benchmarks is 8.90%.

4. Summary and Conclusions

For those conventional networks which are extremely unsuitable for pseudo-exhaustive test, a partitioning tool, Autonomous, was proposed to partition a network into sub-networks so that each sub-network is more suitable for pseudo-exhaustive test. The average number of untestable faults plus dropped faults after network partition is 27.96% less than that of before network partition and is 39.59% less than that of "STG" method. When used with BISTSYN on a set of benchmarks, Autonomous is feasible for very large designs and the fault coverage and fault efficiency are improved. The experimental results show that the average fault coverage of the BISTSYN is 98.35% and 98.81% before and after network partition respectively and that both "Embedding BIST" method is 96.75%. The average fault coverage of the BISTSYN is 99.68% and 99.72% before and after network partition respectively and that of "Embedding BIST" method is 98.07%.

5. References