数理モデル化と問題解決　30－2
（2000．6．26）

部分的にマージ可能なコアを用いた LSI 設計情報保護の数理モデル

伊達 博[†] ヴィクラム アイヤンガー[††] クリシュナンデュ チャクラバーティ[††] 杉原 真[†††]

概要
本論文では，コアを用いて LSI を設計する際に，コアの設計情報を保護するための数理モデルを提案する．最初に，コア分割問題を定義し，それを解くためのアルゴリズムを提案する．設計者により，コアの設計情報の内で保護しなければならない部分 (IP 保護部) と他のコアとマージする部分が指定される．このとき，本アルゴリズムを用いると，IP 保護部を 100 %保護することができ，マージされるコアに対するテスト生成を簡単化することができる．IP 保護部に対して，組込み自己テスト (BIST: Built In Self Test) を適用することにより，コアの内部信号線における可制御性と可観測性とが向上し，テスト時間も短縮することができる．

Mathematical Modeling of Intellectual Property Protection Using Partially-Mergeable Cores

Hiroshi DATE[†], Vikram IYENGAR[††], Krishnendu CHAKRABARTY[††] and Makoto SUGIHARA[†††]

Abstract
We present a mathematical modeling of an intellectual property protection using partially-mergeable cores for core based LSIs. Also, we define a core partitioning problem, and propose an algorithm to solve it. Our technique guarantees 100% protection of critical-IP part of a core, while simplifying test generation for the merged core logic. The controllability and observability of internal lines in the core is enhanced, and test application time is reduced, since the critical-IP part is tested using BIST.

1. Introduction

System design using intellectual property (IP) circuits, known as embedded cores, is the new paradigm in integrated circuit design today. Embedded cores are pre-designed functional blocks that facilitate design reuse, allow complex functionality to be embedded easily in system-on-a-chip (SOC) designs, and lead to shorter product development cycles. As a result, the multi-billion dollar SOC market, which is growing at an annual rate of 60% is a powerful driving force in the electronics industry.

However, while embedded cores have become popular as a means for providing rich functionality within a short design cycle time, core-based systems have greatly exacerbated the testing problem. In order to protect IP for a core, core vendors conceal structural information about their cores and provide only a set of test patterns to be applied that guarantees a specific fault coverage. Thus, core integrators often cannot access the internal logic of these cores to insert design-for-testability hardware to ease test application from surrounding logic. In addition, core testing is difficult since IP cores are often surrounded by user-designed logic (UDL), or embedded within several levels of hierarchical logic. Test sets must be transmitted in their entirety from the integrated circuit pins to these inner layers, often without introducing extra patterns into these test streams, and while conforming to stringent test schedules for all cores in the system.

The IEEE P1500 standard that is being developed for SOC test advocates use of a standardized test wrapper to ease the delivery of test patterns through a test access mechanism (TAM)[7]. Test wrappers must ensure a variety of operation modes, including normal operation, core test mode, interconnect test, detach mode, and bypass mode[8]. In addition, test wrappers need to be able to perform test data serialization if the bandwidth of the TAM is not sufficient; therefore, wrappers contain built-in boundary scan cells at all core terminals. The TAM provides means for on-chip data transport[8]. It can be used to transport test patterns from a pattern source, such as in[2),6)], to a circuit under test (CUT), and to transport test responses from the CUT to a response monitor. A number of test access architectures have been proposed[8].

The complex problem of testing large IP cores can be alleviated if the core provider makes parts of the core mergeable with the UDL. This reduces the IP portion of the core for which precomputed test patterns and dedicated test structures must be provided, as well as gives the core integrator the opportunity to test the remaining non-IP part with the UDL, facilitating system-level and hierarchical tests. However, the core vendor must be provided with the means to protect the critical-IP blocks, while making the core mergeable. There is a fundamental dichotomy between the amount of IP that can be protected by the core vendor and the ease with which an IP core can be tested by the core integrator. Complete IP protection neccessitates black-box testing whereby the IP core cannot be merged with the rest of the logic in the system. On the other hand, totally mergeable cores provide little IP protection for the core vendor.

A procedure to separate some of the low-IP gates in a core was proposed recently in[3]. In this scheme, circuit traversal of the gate-level model of an IP core is carried out by the core vendor to determine which gates in the core can be tested entirely using only the scan chains. These gates can then be removed from the netlist supplied to the core user. However as shown in[3], in a number of cases, the percentage of IP protected (the fraction of the gates removed from the circuit) is very low. The technique further proposed the addition of a small number of boundary scan cells around selected core I/Os to increase the number of gates that can be removed. However, the method in[3], does not guarantee that any particular module, if marked by the

† (財) 九州システム情報技術研究所
Institute of Systems and Information Technologies / Kyushu
†† デューク大学
Department of Electrical and Computer Engineering Duke University
††† 九州大学
Department of Computer Science and Communication Engineering Kyushu University

designer as critical-IP, will actually be protected.

In this paper, we present a novel technique to separate critical-IP blocks from the low-IP part of the core. *Critical-IP* blocks are defined as those modules whose internal structure is unconditionally required to be hidden from the core user. *Low-IP* blocks, on the other hand, refer to the modules and gates whose internal structure can be made known to the core user. For example, critical-IP elements include complex co-processors and ALUs, while low-IP elements include arrays of buffers, multiplexers and similar interface logic. Critical-IP blocks are guaranteed to be protected 100% by our technique. Moreover, our technique also separates out and therefore protects a large portion of the low-IP blocks. The core provider adds a built-in self test (BIST) wrapper to test the IP-protected part of the core. Since the core provider has complete access to the gate-level model of the core, test point insertion for BIST can readily be done before the core is provided to the user. Making an entire core BIST-ready makes too much demands on the core provider. However, the core provider is more likely to add BIST hardware if it provides IP protection to a large portion of the design. In this way, BIST provides an additional return on investment. The resulting partially-mergeable core simplifies the core integrator's problem of transporting a large precomputed test set to the core I/Os, since the critical-IP is tested using BIST. The low-IP part of the core can be integrated with the UDL, which makes testing it more flexible, as well as reduces the amount of ATPG required for the core.

The organization of this paper is as follows. In Section 2, we present the test application methodology for the partially-mergeable core. Section 3 describes the proposed circuit partitioning technique and Section 4 demonstrates the feasibility of our method using the ISCAS 85 benchmark circuits.

2. Test application methodology

The objectives of our core partitioning methodology are:

- Simplify the core integrator's problem of transporting a large precomputed test set to the core I/Os, by integrating certain low-IP blocks of the core with the UDL. In this manner, a portion of the core can be tested along with the UDL using system-level and hierarchical tests.
- Protect critical-IP (as stipulated by the core designer) by separating it from the rest of the core and testing it using BIST. This again means that the core integrator need not apply a precomputed test set for the entire core. TAM design is therefore simplified.
- Reduce the amount of ATPG required for the core, since the critical-IP blocks are tested using BIST.
- Make both critical-IP and low-IP portions of the core more controllable and observable than the entire core. This is achieved since each portion is smaller in size than the entire core, yet the number of primary inputs to each is increased. BIST is therefore made easier for the critical-IP part of the core.

We now describe the proposed core test integration methodology for the critical-IP and low-IP parts of the core. Figure 1 illustrates an embedded core with n functional inputs and m functional outputs that can be partitioned into critical-IP (A) and low-IP (B) parts. Circuit partitioning for complex circuits has been widely used in the past to reduce test generation costs and enhance the controllability and observability of the inputs and outputs associated with the circuit partitions[1].

Figure 2 illustrates an overview of the proposed core test integration methodology. The embedded core with n functional inputs and m functional outputs from Figure 1 is partitioned into part A with n_1 functional inputs and m_1 functional outputs, and part B with n_2 functional inputs and m_2 functional outputs, respectively; see Figure 2. Part A containing the critical-IP blocks is surrounded by a BIST wrapper similar to the wrapper advocated by the IEEE P1500 standard, with the boundary scan cells configured as LFSR/MISR. Figure 3 presents a detailed view of the test architecture. The two-mode test methodology is as follows. In the first mode of test application (mode 1), the LFSR supplies test patterns to the inputs of A, the protected-IP part of the core. Structural information about A need not be supplied to the core integrator, and is therefore guaranteed to be protected. Part B, which represents low-IP, may be merged with the UDL by the core integrator for the purposes of ATPG in mode 2. In this manner, a bus-based TAM is not required to supply precomputed test patterns, even for the critical-IP part.

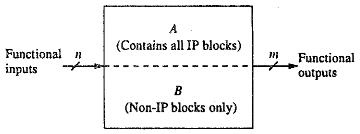

Figure 1 An embedded core partitioned into critical-IP (A) and low-IP (B) parts.

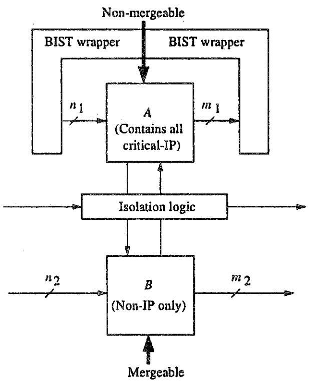

Figure 2 The proposed test methodology based on core partitioning.

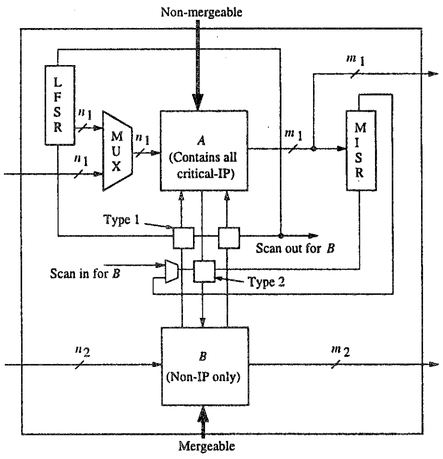

Figure 3 Detailed view of the proposed core test integration architecture.

Isolation cells are used to separate A from B (Figure 3). These cells are inserted by the core provider for testing. There are two types of isolation cells used. Type 1 cells lie on the internal lines that lead from B to A. These have three modes of operation: (i) transparent in normal operation, (ii) connected to LFSR for pseudorandom testing of A (mode 1), and (iii) output scan cells for B (mode 2). The second type of isolation cells lie on the internal lines that lead from A to B. These also have three modes of operation: (i) transparent in normal operation, (ii) connected to the MISR for pseudorandom testing of A (mode 1), and (iii) input scan cells for B (mode 2). The hardware overhead of our approach is directly related to the number of isolation cells inserted into the design. Therefore, minimizing the number of the isolation cells is an important objective. In the next section, we describe our method to efficiently separate the critical-IP blocks from the low-IP part of the core, while minimizing the number of isolation cells.

3. Core partitioning

The specific problem that we address while partitioning the core is stated as follows:

- P : Given a netlist representation of the core at the functional level with certain blocks marked by the designer as critical-IP, determine a partition of the core into disjoint logic blocks A and B, so as to (i) ensure that all critical-IP blocks appear in A, (ii) minimize the number of internal circuit lines connecting A and B, (iii) ensure a maximum limit l on the LFSR size (number of core primary inputs that form inputs to A), and (iv) ensure that no isolation cells lie on critical paths.

This problem can be proved to be NP-complete by restriction to a special case. Consider a special case of the problem P containing only one critical-IP block. Construct a graph $G(V, E)$ that contains a vertex $v \in V$ corresponding to each primary input, primary output, gate or functional block in the core. The set of directed graph edges E represents internal lines between functional blocks. Assign a weight of ∞ to each edge on a critical path, and an integer weight of $w(e)$ to each other edge e that lies between vertices v_i and v_j, where $w(e)$ is the width of the bus that connects the corresponding modules i and j in the netlist. Label the vertex corresponding to the critical-IP block as vertex s, and any one vertex corresponding to a low-IP block as vertex t. Now, consider the following "Minimum cut into bounded sets decision problem" from[4]: "Is there a partition of V into disjoint sets V_1 and V_2 such that $s \in V_1, t \in V_2, |V_1| \leq S, |V_2| \leq S$, where $1 \leq S \leq |V|-1$, and such that the sum of the weights of the edges from E that have one endpoint in V_1 and one endpoint in V_2 is no more than K". The *minimum cut* problem is NP-complete[4], therefore the problem P that we are addressing, of which a special case is equivalent to minimum cut, is also NP-complete.

We have developed a search algorithm based on several heuristics to solve the core partitioning problem efficiently. Our algorithm is illustrated in pseudocode in Figure 4. The core netlist is transformed into a graph $G(V, E)$ as described earlier; and the vertices corresponding to critical-IP blocks in the core are marked. If there is a critical path edge lying between a marked and unmarked vertex, the unmarked vertex is marked in order to protect the critical path edge from being cut. Forward breadth-first traversal is performed from each marked vertex to mark all edges and vertices lying in its output cone. Backward traversal from each critical-IP vertex is then performed to determine the set of primary input vertices that can reach the critical-IP vertices. These inputs are the *candidate* primary inputs for A.

Next, we determine which of the candidate inputs, if selected as an LFSR input to A would minimize the

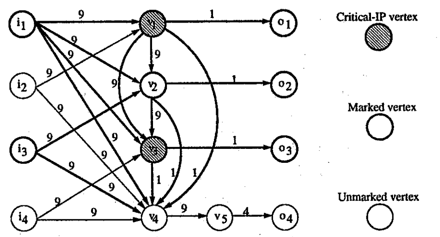

Figure 5 The graph representation of the circuit c432 from the ISCAS 85 benchmark suite.

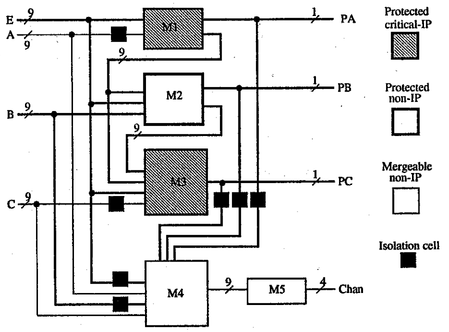

Figure 6 The proposed core partitioning algorithm applied to c432.

number of edges between partitions V_1 and V_2, by maximizing the number of marked vertices in part V_1. This is determined by a heuristic *value* that is assigned to each candidate primary input. This value corresponds to the number of critical-IP vertices reachable from it. The candidate input with the highest value is selected as an LFSR input to A and marked. Traversal from the selected input vertex is performed and all edges and vertices lying in its output cone are marked. In this manner, after l primary inputs have been chosen to be connected to the LFSR, the algorithm halts, and all edges lying between a marked and unmarked vertex are cut. Since the complexity of traversal is linear in the number of vertices and edges in the graph, our method is not computationally expensive. The marked vertices now correspond to the blocks in A, while the unmarked vertices correspond to B.

Figure 5 illustrates the graph representation of the ISCAS 85 circuit c432 with marked edges and vertices shown in bold. The width of the buses connecting modules in the circuit are mapped to weights on the edges in the graph. Figure 6 shows the circuit partitioned according to the proposed algorithm. The circuit has five high-level modules, of which M1 and M3 are chosen as critical-IP. The modules and buses shown in bold correspond to the marked edges and vertices in the graphical representation of the netlist. Primary input buses E and B are chosen to be connected to the LFSR. Modules M1, M2 (low-IP) and M3 are protected, while modules M4 and M5 are merged with the UDL.

4. Experimental results

In this section, we present experimental results of our technique for several high-level representations of ISCAS 85 circuits[5]. We performed experiments on core partitioning at the functional-level rather than at the gate-level since partitioning complex cores to protect critical-IP is more meaningful when large blocks, rather than individual gates, in the core are treated as IP modules. For our experiments, we chose the most

```
Procedure CORE_PARTITION()
/* Partition a given core netlist into critical-IP (part A) and low-IP (part B)*/
begin
1.     mark all critical-IP nodes in the graph;
2.     for (each critical path edge e)
3.         if (e goes from unmarked node b to marked node a) then
4.             mark node b;
5.     for (each marked node a) /* forward traversal from marked nodes */
6.         mark all edges leaving a;
7.         for (each node c in fanout of a)
8.             if (c is marked) then
9.                 continue traversal from c;
10.            else /* if c is unmarked */
11.                if (each incoming edge to c is marked) then
12.                    mark c;
13.                    if (there is a critical path edge between c and another unmarked node d) then
                       /* ensure that critical path edges are not cut */
14.                        mark d and continue traversal from c;
15.                else
16.                    halt traversal at c and continue traversal from fanout of a;
17.     perform backward traversal from each critical-IP node towards primary input nodes;
18.     label each primary input node reached as a candidate;
    /* value of each candidate = number of critical-IP nodes that can reach it */
19.     label each candidate with its value;
20.     LFSR_size = 0;
21.     while (LFSR_size < k)
22.         LFSR_size = LFSR_size + 1;
23.         mark unmarked candidate d having highest value;
24.         perform forward traversal from d to mark its output cone; /* steps 5 to 16 */
25.     label each edge lying between a marked node and an unmarked node as cut;
    /* the marked nodes form partition A and unmarked nodes form partition B */
end
```

Figure 4 Algorithm for core partitioning that guarantees 100% protection of critical-IP blocks.

Core	Number of primary inputs n	Number of primary outputs m	LFSR size	Number of isolation cells
c432	36	7	27	30
c499	41	32	32	8
c880	60	26	42	32
c1355	41	32	32	8
c1908	33	25	19	17
c2670	233	140	17	37
c3540	50	22	14	32
c5315	178	123	36	20

(a) LFSR size and number of isolation cells

Core	Total number of modules	Number of designated critical-IP modules	Number of modules protected
c432	5	2	3
c499	2	1	1
c880	6	3	3
c1355	2	1	1
c1908	7	4	5
c2670	10	2	6
c3540	10	1	5
c5315	10	2	4

(b) The number of modules protected

Table 1 Experimental results on core partitioning for IP protection

complex of the functional blocks as critical-IP, and ensured that the blocks chosen as critical-IP were as far apart topologically in the netlist as possible. This provides "worst-case" instances of the partioning problem.

Table 1 presents an initial set of experimental results of our method for several ISCAS 85 circuits. We treat the high-level models of these circuits as cores and designate one or more functional blocks in the core as critical-IP blocks. Table 1(a) presents experimental data on the LFSR size and the number of isolation cells for the ISCAS 85 circuits. The LFSR does not contribute to hardware overhead since it can be constructed in the form of a partial P1500 test wrapper, which would otherwise have been required for test data serialization and interfacing with the test bus, if the entire core were tested as a black-box. In Table 1(b), we list the number of modules in the high-level models of the circuits, the number of modules designated as critical-IP, and the number of modules protected by the proposed partitioning method. Note that the number of modules protected includes 100% critical-IP protection.

5. Conclusion

We have presented a novel technique for making embedded cores partially-mergeable. The proposed method guarantees 100% protection of critical-IP, while allowing the core integrator to merge low-IP parts of the core with the UDL for the purposes of testing. Test generation is simplified since the amount of logic to be tested in the critical-IP part of the core is now smaller than the entire core and hierarchical and system-level tests can be used for the low-IP part by the core integrator. Testing time is reduced significantly because the critical-IP is tested at-speed using BIST. Furthermore, controllability and observability of both parts A and B are increased. Experimental results on several ISCAS 85 circuits demonstrate the usefulness of our technique in providing 100% critical-IP protection while greatly simplifying the testing problem at a reasonable hardware overhead.

References

1) M. Abramovici, M. A. Breuer and A. D. Friedman, *"Digital Systems Testing and Testable Design,"* Computer Science Press, New York, NY, 1990.
2) K. Chakrabarty, B. T. Murray and V. Iyengar, "Built-in pattern generation for high-performance circuits using twisted ring counters," *Proc. VLSI Test Symposium*, pp. 22-27, 1999.
3) K. De, "Test methodology for embedded cores which protects intellectual property," *Proc. VLSI Test Symposium*, pp. 2-9, 1997.
4) M. S. Garey and D. S. Johnson, *"Computers and Intractability: A Guide to the Theory of NP-Completeness,"* W. H. Freeman and Company, New York, 1979.
5) M. C. Hansen, H. Yalcin and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Design & Test of Computers*, vol. pp. 72–80, July-September 1999.
6) V. Iyengar, K. Chakrabarty and B. T. Murray, "Deterministic built-in self testing of sequential circuits using precomputed test sets," *Journal of Electronic Testing: Theory and Applications*, vol. 15, pp. 97–114, August 1999.
7) E. J. Marinissen, Y. Zorian, R. Kapur, K. Taylor and L. Whetsel, "Towards a standard for embedded core test: An example," *Proc. International Test Conference*, pp. 616–627, 1999.
8) Y. Zorian, E. J. Marinissen and S. Dey, "Testing embedded-core based system chips," *Proc. International Test Conference*, pp. 130–143, 1998.