Doctoral Dissertation

Studies on DFT for Reducing Its Area and Test Application Time of System-on-a-Chip

Masahide Miyazaki

February 2, 2006

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Masahide Miyazaki

Studies on DFT for Reducing Its Area and Test Application Time of System-on-a-Chip

Masahide Miyazaki

Abstract

With the progress of the semiconductor process technology, a lot of functions came to be included in a System-on-a-Chip (SoC). To test such SoCs, a test pattern is prepared for each core, and the modular testing of embedded cores is carried out. For a core which is reused in a high-level design methodology, the designer has to determine the design for test (DFT) method along with the required quality and cost. In addition, many memories with different sizes and frequencies are being used in SoCs. To test such memories, we need memory BIST techniques. However, if memory BIST logics were individually added to these various memories, the area overhead would be very high.

This thesis proposes a DFT selection method for reducing test application time. At first an SoC test architecture generation framework is presented. It contains a database, which stores the test cost information of several DFTs for every core, and a DFT selection part that performs DFT selection for minimizing the test application time using this database in the early phase of the design flow. Moreover, the DFT selection problem is formulated and the algorithm to solve this problem is proposed.

This thesis also proposes a systematic way of memory grouping, for reducing area by sharing BIST logic by a group. Two types of memory connection method and the compatibility graphs that indicate whether the connection of each memory is possible are introduced. The compatibility graph of each type of the connection makes it possible to search for minimizing the area under constraints of the maximum test application time and maximum power consumption. Furthermore, a memory-grouping problem is formulated and an algorithm to solve the problem is proposed.

The effectiveness of all the above methods has been demonstrated through experimental results.

keywords：

SoC, test scheduling, test access mechanism, wrapper, design for test, memory BIST

List of Publications

Journal Papers

1. Masahide Miyazaki, Toshinori Hosokawa, Hiroshi Date, Michiaki Muraoka and Hideo Fujiwara, “A DFT selection method for reducing test application time of system-on-chips”, IEICE Transactions on Information and Systems, Vol. E87-D, No. 3, pp.609-619, Mar. 2004.
2. Masahide Miyazaki, Tomokazu Yoneda and Hideo Fujiwara, “A Memory grouping method for reducing memory BIST logic of system-on-chips”, IEICE Transactions on Information and Systems, Vol. E89-D, No. 4, pp.1490-1497, April 2006.

International Conference Papers (Reviewed)

1. Masahide Miyazaki, Toshinori Hosokawa, Hiroshi Date, Michiaki Muraoka and Hideo Fujiwara, “A DFT selection method for reducing test application time of system-on-chips”, Proceedings of the IEEE 12th Asian Test Symposium, pp.412-417, Nov. 2003.
2. Masahide Miyazaki, Tomokazu Yoneda and Hideo Fujiwara, “A memory grouping method for reducing memory BIST logic of system-on-chips”, Proceedings of the IEEE 6th Workshop on RTL and High Level Testing, pp.31-37, July 2005.
3. Masahide Miyazaki, Tomokazu Yoneda and Hideo Fujiwara, “A Memory grouping method for sharing memory BIST logic”, Proceedings of the IEEE 11th Asia and South Pacific Design Automation Conference, pp.671-676, Jan. 2006.

Contents

List of Figures

List of Tables

1 INTRODUCTION

1-1 Motivation

With the progress of the semiconductor process technology, the gate count of System-on-Chip (SoC) is increasing as large as one hundred million gates through the use of 90 nm process design rule toward 2010. As the size of the SoC is getting larger, the reduction of the design productivity will be the most important issue. The technologies that solve this issue are the design reuse methodology and design automation at the high-level design phase. Research and development of these technologies is the key to innovate the SoC design methodology.

In order to reduce design time, SoCs consist of a large number of reusable cores. To test such SoCs, a test pattern is prepared for each core, and the modular testing of embedded cores is carried out. For a core, which is reused in a high-level design methodology, the designer has to determine the design for test (DFT) method along with the required quality and cost. For this reason, the technique of determining an SoC test architecture including DFT selection of each core, taking test cost and test quality into consideration, during the early design phase is needed.

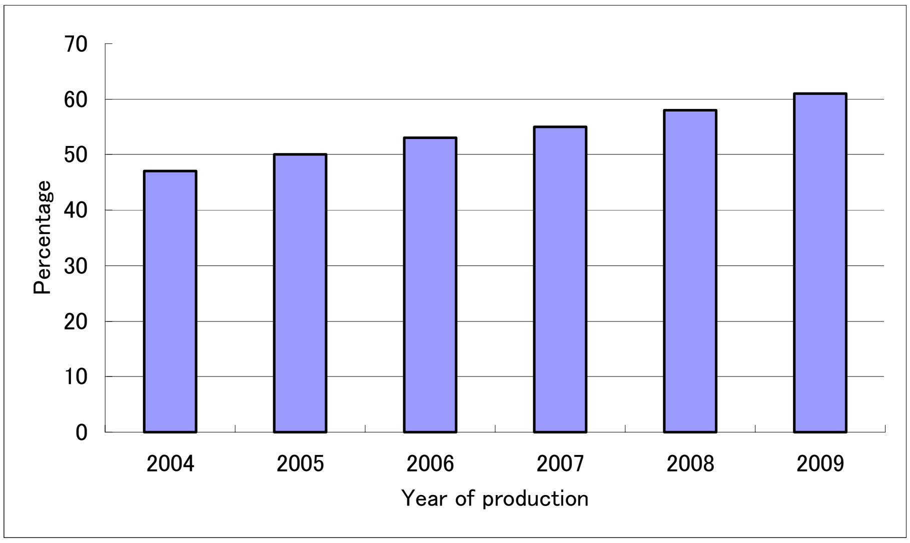

Fig. 1-1 Percentage of memory in area of SoC

In addition, with the increasing demand for SoCs to include rich functionality, SoCs are being designed with hundreds of small memories with different sizes and frequencies. If all the memories are tested from ATE, they have to use interface sequentially. It takes long test application time so that it is not acceptable. Therefore, memory BIST technique must be used to test memories on SoCs. On the other hand, if memory BIST logics were individually added to these various memories, the area overhead would be very high. Moreover, the percentage of embedded memory used in SoC designs is steadily increasing. According to the prediction of the International Technology Roadmap for Semiconductors 2004 update [25], the area of the memory reaches 60% of SoC in 2009 (Figure 1-1). Therefore, it is easily surmisable that the area of memory BIST logic added to the memory grows, too. For this reason, memory BIST logic sharing technique that can ease an increase in BIST logic by testing two or more memories by a couple of BIST logic is very important. But the increase of the number of memories makes it difficult to decide groups of memories, which share BIST logic. So, a systematic way of memory BIST logic sharing is needed.

1-2 Thesis Organization

This thesis is organized as follows.

Chapter 2 is devoted to DFT selection method for reducing test application time. At first it proposes a framework of SoC test architecture generation. Thereafter, core's DFT method and the precondition of this research work are described. After that, the formulation of the DFT selection problem and an algorithm are proposed. It is followed by experimental results to show the effectiveness of this method.

Chapter 3 is devoted to memory grouping method for reducing area of memory BIST logic. At first two types of memory-connection methods for memory BIST wrapper sharing are presented. Thereafter, the memory-grouping problem formulation and an algorithm to solve the problem are presented. In addition, the effectiveness of this technique is demonstrated experimentally.

Finally, this thesis concludes with Chapter 4 where the main accomplishments of the work are outlined and the future directions are identified.

1-3 Contribution of This Thesis

This thesis makes following basic contributions.

First, for reducing test application time of System-on-Chips, this thesis proposes a framework that includes the test cost estimation step and test cost information database. In the test cost estimation step, information on each core is estimated, and the result is output to the test cost information database. In addition, for each core, the test cost information of two or more DFTs are estimated so that the most suitable DFT can be selected.

Second, this thesis proposes a systematic way of memory group decision using graph theoretic approach for determining memory group that share BIST logic. It is necessary to connect memories to share memory BIST logic, but there are two or more types of the connection, and the area, the power consumption and the test application time depend on the type of the connection. To achieve a good solution, we have to optimize the type of the connection. This thesis treats two types of connection methods, and two types of compatibility graphs that indicate the connectivity are introduced. The search to minimize the area under constraints of the test application time and power consumption becomes possible by using the compatibility graph of each type of the connection.

2 A DFT SELECTION METHOD FOR REDUCING TEST APPLICATION TIME OF SYSTEM-ON-CHIPS

2-1 Introduction

Effective modular test requires efficient management of the test resources for core-based SoCs. This involves the design of core test wrappers and TAMs (Test Access Mechanisms), and the scheduling of core tests. In recent years, many research works relevant to these issues have been presented.

Core test wrapper design and TAM design are important since they have impact on hardware overhead and test application time. There are three main approaches to achieve accessibility of embedded cores. The first approach is based on test bus architectures, where the cores are isolated from each other in test mode using a dedicated bus [1][2][3] around the cores to propagate test data. The second approach uses boundary scan architectures [4][5] to isolate the core during test. The third approach uses core bypass mode [6] or transparency [7][8][9]. Wrapper and TAM design include wrapper optimization, core assignment to TAM wires, sizing of the TAMs, and routing of TAM wires. So, wrapper and TAM co-optimization approach [10] is one of the important subjects in modular testing.

The objective of test scheduling [11][12][13] is to minimize test application time under one or more of the following constraints: maximum TAM width and maximum allowed power consumption. Furthermore, optimal wrapper width selection and test scheduling techniques have been proposed [14][15].

Most of the above research works assume scan design as a core's DFT, or do not mention about a core's DFT. To cope with the testing of large and complex SoCs, the modular testing of embedded cores will have to be rethought. For a core, which is reused in a high-level design methodology, the designer has to determine the DFT method along with the required quality and cost. For this reason, the technique of determining an SoC test architecture including DFT of each core, taking test cost and test quality into consideration, during the early design phase is needed.

In this chapter, we present the DFT selection method for reducing test application time under the following constraints: maximum TAM width, maximum allowed power consumption, total area size, and test data size. Each core's DFT is chosen from scan design or non-scan DFT [16][17].

This chapter is organized as follows. In section 2-2, a framework of SoC test architecture generation is proposed. In section 2-3, core's DFT method and the precondition of this research work are described. In section 2-4, the formulation of the DFT selection problem and an algorithm are proposed. It is followed by experimental results in section 2-5. Finally, section 2-6 concludes this chapter.

2-2 A Framework of SoC Test Architecture Generation

In a high level design methodology, the designer has to determine the DFT method of each core along with the required quality and cost. So we propose a new framework that includes DFT selection phase in the SoC early design phase. Figure 2-1 shows our SoC test architecture generation framework. The framework consists of the following stages.

In the first stage, the test cost information on each core is estimated, and the result is output to the test cost information database. In addition, for each core, the test cost information of two or more DFTs are estimated. Test cost information includes the following information.

- Test application time
- TAM width
- Power consumption
- Area size
- Test data size

In a SoC design, a great portion of it is filled with reused IPs. Then, actual test cost information obtained from a past design can be used in a reused portion. Moreover, if possible, an estimation of the test cost information for a new design can be made based on the test cost information of the past design. For example, when there is an actual result value of the scan design of a certain core, it is possible to make an estimation of the test cost information in which the number of the scan chain was changed.

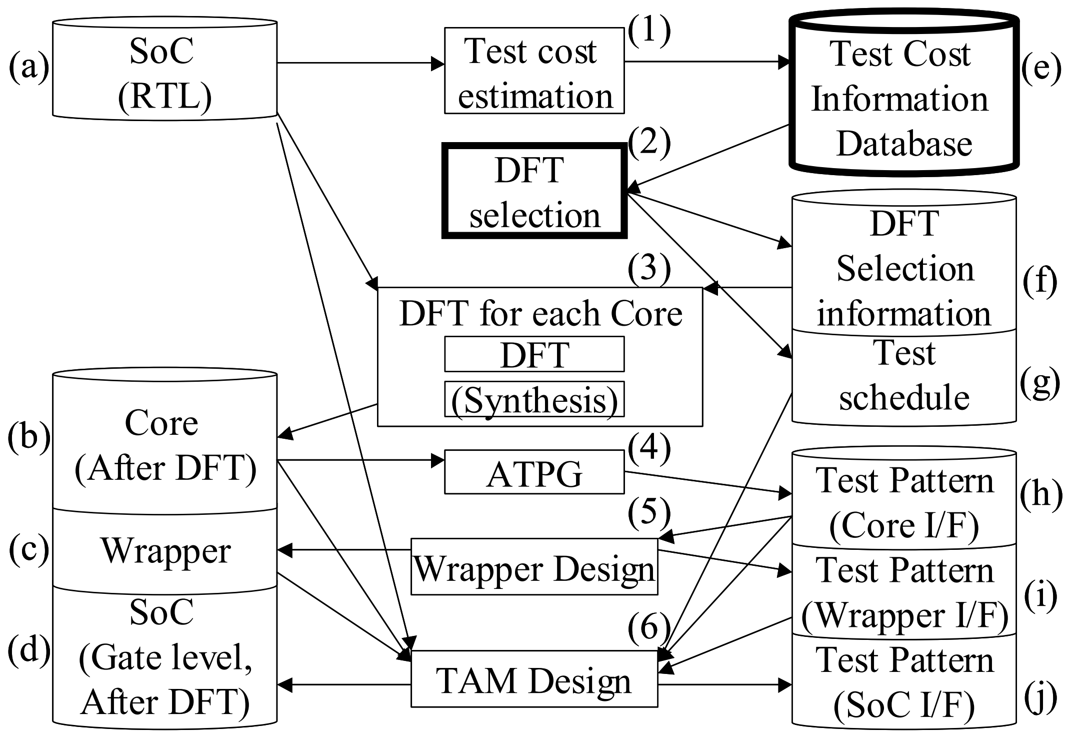

Fig. 2-1 Framework of SoC test architecture generation

If high accuracy is needed, it is necessary to carry out logic synthesis according to product specification, and to actually perform some scan design. If high accuracy is needed, it is necessary to carry out logic synthesis according to product specification, and to actually perform some scan design. If high accuracy is needed, it is necessary to carry out logic synthesis according to product specification, and to actually perform some scan design. If high accuracy is needed, it is necessary to carry out logic synthesis according to product specification, and to actually perform some scan design. If high accuracy is needed, it is necessary to carry out logic synthesis according to product specification, and to actually perform some scan design. However, if accuracy is not needed, some variations will be created only reflecting the change of the scan chain length when changing the number of scan chains by assuming that the area size, the number of test patterns and the number of flip-flops do not change. In this case, test application time, TAM width, and test data size are easily calculated. It is difficult to estimate the value of power consumption with high accuracy. However, there are conventional tools, which are able to estimate power consumption for an RTL description, and it is easy to perform relative comparison among two or more DFT(s).

Newly designed cores need to carry out data creation by actually applying DFT using RTL (a).

However, accumulating the actual result value in the test cost information database (e) reduces the cost required to estimate test cost information, and it leads to an increase in accuracy.

In the second stage, each core's DFT selection is optimized to reduce the total test application time using test cost information (e), which was estimated in the first stage, and each core's selected DFT (f) and test schedule (g) are output. Test schedule means test start time and test end time of each core.

According to the DFT selection information of each core, design for test of each core is performed in the third stage. The test pattern of the core (h) is created in fourth stage using existing ATPG tools.

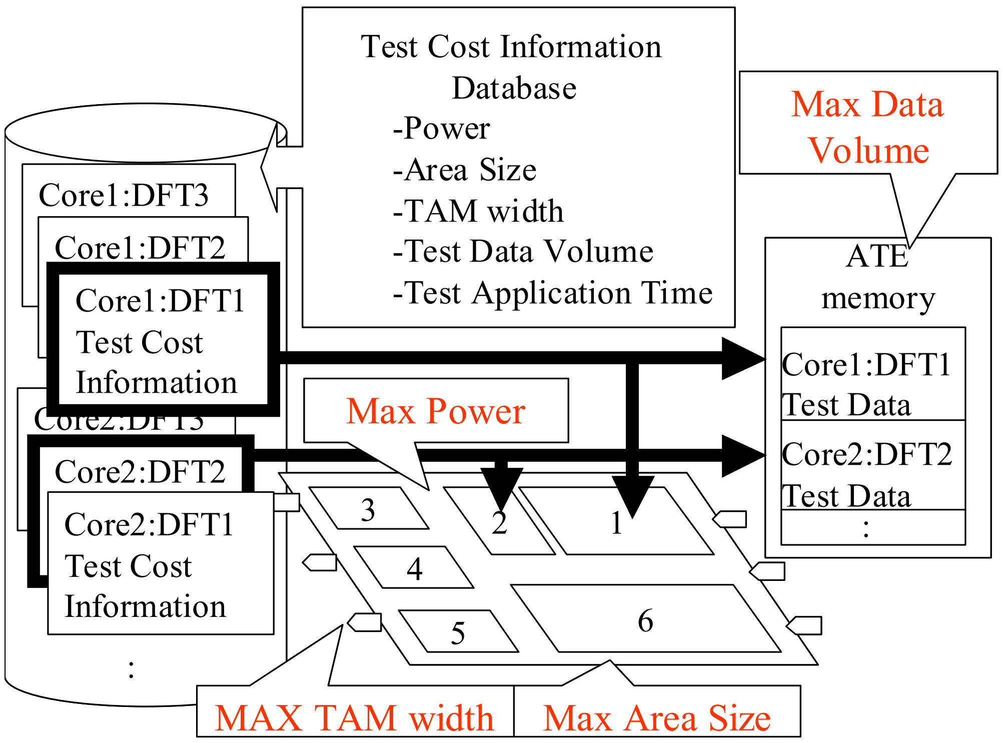

Fig. 2-2 DFT selection

In the fifth stage, the core's test wrapper is designed. Based on the test pattern of a core, which was created in the third stage, the bit width compression function is incorporated if needed. Moreover, the test pattern is modified to match the interface of the designed wrappers. In the last stage, the cores tested simultaneously are divided into several TAMs according to the test schedule (h), which was created in the second stage, and SoC design that include TAMs is created. Moreover, the test pattern of each core is edited and output with the interface of SoC pin.

The DFT selection stage is especially important among the above mentioned framework stages. Figure 2-2 shows the work for which a designer is asked in this stage. The designer determines the SoC's test strategy, including DFT selection of each core to reduce test application time under the following constraints: maximum TAM width, maximum allowed power consumption, total area size, test data size.

2-3 DFT of Each Core

2-3-1 Scan Design Method

Full scan design is one of the most popular DFT methods. The scan test application time depends on the maximum scan chain length. As a large TAM width can be taken, a scan chain can be divided and thus the maximum scan chain length and test application time can be shortened.

Figure 2-3 shows an example of the relationship between scan chain length and number of chains in the case in which the number of FFs is 100. The vertical axis expresses the scan chain length in logarithm scale, and the horizontal axis expresses the number of scan chains. Figure 2-3 shows that when the number of scan chains is large, the scan chain length function becomes a stair function. Therefore even if the TAM width increases, the test application time is not always shortened [14].

In modular testing, the TAM width of each core should be selected to minimize the total test application time. However, in the stair function portion shown in Figure 2-3, it is enough that only the pareto-optimal-points [14] are taken into consideration as the candidate points of the selection.

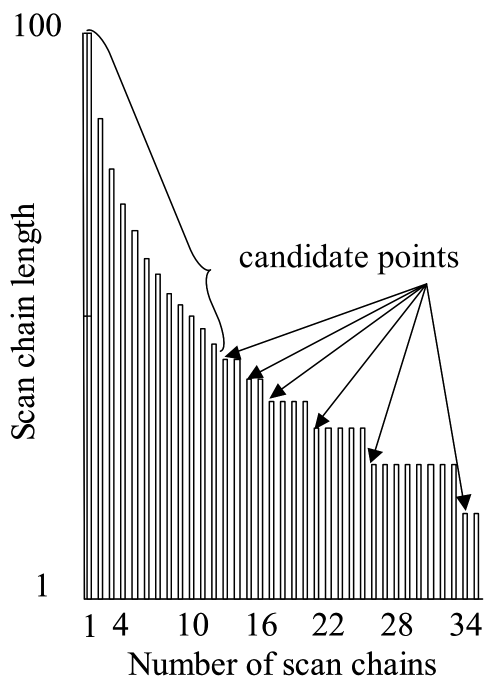

Fig. 2-3 Relationship between scan chain length and number of chains

2-3-2 Non-Scan DFT Method

Scan design methods have the following disadvantages concerning test cost and test quality:

- The additional test circuits for DFT cause the degradation of performance.
- The test length is very long.
- It is not suited for at-speed-testing.

In order to drastically improve the above-mentioned disadvantages while keeping complete fault efficiency, non-scan DFT methods [16][17] for RTL design circuits were proposed. In this thesis, the non-scan DFT method (NS-DFT) of reference [16] shall be chosen as another DFT of a core.

NS-DFT needs parallel access from LSI pins to all the inputs and all the outputs. Thus, NS-DFT is inapplicable to a core with input and output larger than the number of LSI pins. Therefore, the core test wrapper with bit width compression function shall be prepared. Figure 2-4 shows the wrapper design. Wrapper mode signals are used to change between five modes: normal, test, isolation, input interconnect test, and output interconnect test. In normal mode, Functional inputs and Functional outputs are connected to the core. When another core is tested, the core inputs are fixed to isolate, if needed. In input interconnect test mode, the Functional Inputs are connected to the Encoder and observed at Test Outputs. In output interconnect test mode, the Functional Outputs are controlled from Test Inputs. In test mode, encoded test patterns are supplied to Test Inputs, and decoded patterns are provided to the core's inputs. The outputs are encoded at Encoder, and observed at Test Outputs. The clock and Asynchronous signal are directly connected in all modes.

The input compression technique shall use the coding technique using EOR network [18][19]. The output compression technique shall use EOR tree. These bit width compression techniques do not change test length, but the area size of the wrapper is different with the compression ratio. If the compression rate is high, the area size of the decoder and the encoder increase. Moreover, if the area size increases, power consumption increases under the same frequency. Since the reduction of TAM width increases the possibility that cores can carry out a simultaneous test, the total test application time may be shortened. Thus, there is a trade-off between the total test application time and area, and the total test application time and power consumption.

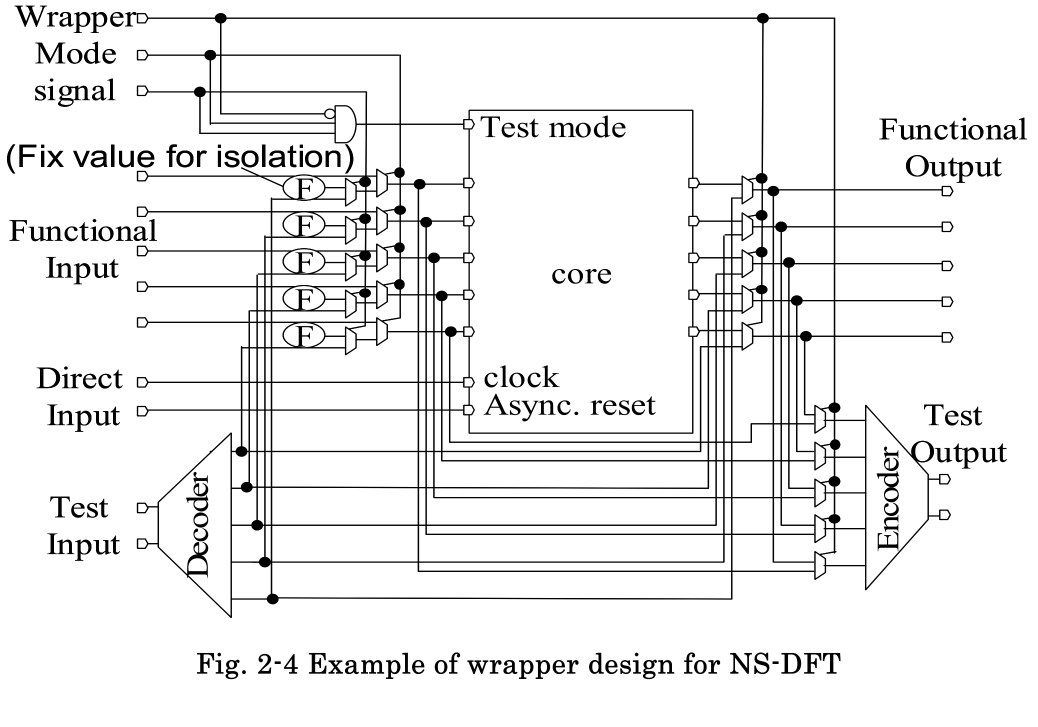

Fig. 2-4 Example of wrapper design for NS-DFT

2-4 DFT Selection Problem Formulation and Algorithm

2-4-1 DFT Selection Problem Formulation

To select the DFT of each core in order to reduce the test application time under constraints, we formulate the DFT selection problem as follows.

Inputs:

(1) Test cost information of each core : D=D_{ij} (DFT$_{ij}$, w_{ij}, p_{ij}, v_{ij}, a_{ij}, t_{ij})

Here, DFT$_{ij}$, w_{ij}, p_{ij}, v_{ij}, a_{ij}, and tij are, respectively:

- DFT$_{ij}$: j th DFT of core i.
- w_{ij}: TAM width of core i to which DFT$_{ij}$ is applied.
- p_{ij}: maximum power consumption of testing core i to which DFT$_{ij}$ is applied.
- a_{ij}: Area of core i to which DFT$_{ij}$ is applied.
- v_{ij}: The amount of test data of core i to which DFT$_{ij}$ is applied.
- t_{ij}: Test application time of core i to which DFT$_{ij}$ is applied.

(2) Maximum TAM width of the SoC : W

(3) Maximum available peak power of the SoC: P

(4) Maximum amount of the total test data size: V

(5) Maximum area size of the SoC: A

Outputs:

(1) Selected DFT of each core

(2) Test schedule

Objective:

The test application time of the SoC is minimum, and constraints (Input: (2), (3), (4), (5)) are satisfied.

To solve this problem, an algorithm is proposed in the next paragraph.

2-4-2 Algorithm

The algorithm is shown in Figure 2-5. The following variables are used in it.

C: Variable to store test cost information of each core: C=C_i (CDFT$_i$, w_i, p_i, v_i, a_i, t_i)

CDFT$_i$, w_i, p_i, v_i, a_i, t_i are, respectively:

```
Procedure DFT_selection(D,W,P,A,V)
Define initial DFT assignment  Cinit;
C=Cinit;
current_tat= sum of  the all core's test application time;
best_tat=rectangle_packaging(C,W,P,A,V,current_tat);
do{
    current_tat= best_tat;
    Ccurrent = C;
    for( i = 1; i <= number of cores; i++ ){
        for( j = 1; j <= number of DFTs of core i; j++){
            if( $CDFT_i$ != $DFT_{ij}$ ){
                $C_i = D_{ij}$   /*change the core's DFT*/
                trial_tat= rectangle_packaging(C,W,P,A,V, current_tat);
                if( trial_tat < best_tat ){
                    best_tat = trial_tat;
                    Cbest = C;
                }
            }
        }
        C =Ccurrent;
    }
    if( best_tat <current_tat ){
        restore C = Cbest;
    }
}while(best_tat<current_tat);
```

Fig. 2-5 DFT selection algorithm

$CDFT_i$: selected DFT of core i.

w_i: TAM width of core i.

p_i: Power consumption of core i.

a_i: Area of core i.

v_i: Test data volume of core i.

t_i: Test application time of core i.

Cinit: Test cost information under the initial DFT selection.

Ccurrent: Variable which stores the test cost information under the present DFT selection.

Cbest: Variable which stores the test cost information under DFT selection of the minimum test application time.

current_tat: Variable which stores the test application time.

best_tat: Variable which stores the minimum test application time.

trial_tat: Variable which stores the test application time.

In the first step, the initial DFT selection and the test cost information *Cinit* are created (line1). Initialize the variable that stores the test cost information current_tat with the sum of the test application time of each core of the initial DFT selection (line2).

Next, test scheduling aiming at minimizing test application time is performed with the rectangle_packaging algorithm [14][15]. The return value of the above-mentioned algorithm is the test application time of the whole SoC. This result is stored in the variable best_tat (line4).

Hereafter, while best_tat is updated, change the DFT selection and test scheduling repeatedly (line5-line24). The loop iteration is as follows.

current_tat is updated with the value of best_tat (line6). Test cost information *C* under the current DFT selection is held as *Ccurrent* (line7). The following procedure is performed on all DFTs of each core (line8-line20).

If the current DFT of core i is not DFT_{ij} then copy D_{ij} to C_i (line10 - 11). Then, test scheduling is performed (line12). Consequently, if the obtained test application time is shorter than best_tat (line13), best_tat will be updated (line14) and the test cost information *C* will be stored as *Cbest* (line15). After all DFTs of the concerned core are tried, *C* is written back to *Ccurrent* (line18). Then, after all core's trial, if best_tat was updated after trying all cores and all DFTs (line21), *Cbest* is copied to *C* (line22), and go back to the line 6 (line24), otherwise the algorithm ends. DFT of each core of *Cbest* obtained at the end is the solution of DFT selection algorithm.

This algorithm performs exhaustive search in the worst case. The complexity of the scheduling algorithm is *O(nlogn)*. The search space size of the DFT selection is $\prod_{i=1}^{n} j$.

Therefore the complexity of this algorithm is $O((n\log n) \times \prod_{i=1}^{n} j)$. In other words, if the number of DFTs of each core is at most *m*, the total complexity is $O(m^n n\log n)$.

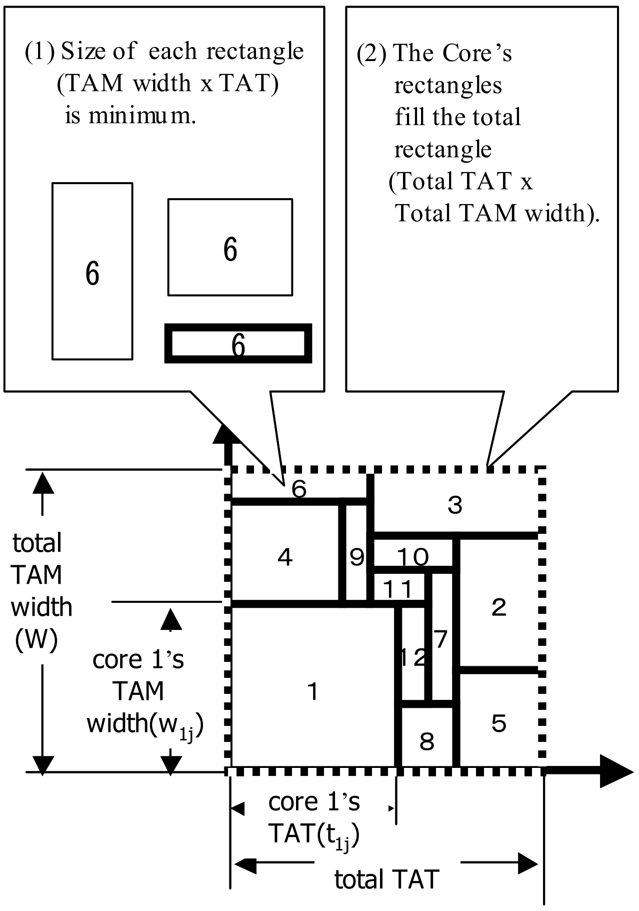
(1) Size of each rectangle (TAM width x TAT) is minimum.
6
6
6
(2) The Core's rectangles fill the total rectangle (Total TAT x Total TAM width).
6
3
4
9
10
11
2
12
7
1
8
5
total TAM width (W)
core 1's TAM width(w_{1j})
core 1's TAT(t_{1j})
total TAT

Fig. 2-6 Lower bound case of test schedule

Because the search space becomes large rapidly to the increase in the number of cores and the number of DFTs, a search space reduction heuristic is needed for practical use.

Generally, in order to reduce the search space, and in order to estimate the quality of a solution, it is useful to know the lower bound of the cost function. In the DFT selection problem, it is difficult to find the true optimal solution. However, it is possible to calculate some lower bounds of the test application time.

One of the optimum cases is shown in Figure 2-6. The numbered rectangles represent the test application time and the TAM width of each core under the selected DFT. The rectangle's vertical length represents the TAM width, and the rectangle's horizontal length represent test application time. The dotted line shows the total rectangle, in which the vertical length represents the total TAM width and the horizontal length represents the total test application time. If (1) the DFT of each core is selected such that the size of each core's rectangle is minimum, and (2) the core's rectangles fill the total rectangles, then the total test application time is the true minimum. Thus a lower bound of test application time lb1 can be calculated as follows.

$$lb1 = \sum_{i=1}^{n} \min_j(t_{ij} \times w_{ij})/W$$

Furthermore, the total test application time cannot be shorter than a core's test application time. Therefore another lower bound lb2 can be calculated as follows.

$lb2 = \max_i (\min_j(t_{ij}))$

Let the largest of these two values lb1 and lb2 be the lower bound LB.

$LB = \max(lb1, lb2)$

If some choices can be deleted so that LB is not changed, it may be used effectively in a heuristics to prune the search space in the DFT selection algorithm. The other usage of LB is that the quality of a solution can be pessimistically estimated by comparing with LB.

2-5 Experimental Results

2-5-1 Experimental Environment

The experimental environment is as follows.

(1) The experiment had been held on the Sun ultra80 workstation, (Sun OS 5.6), 400MHz, 2Gbyte memory.

(2) The proposed algorithm was implemented in C.

(3) Nine RTL designs were used as experimental circuit.

(4) To prepare the test cost information, we used the following conventional EDA tools:

power consumption estimation : Wattme / Artgraphics

logic synthesis : DesignCompiler / Synopsys

Scan path synthesis :DFT Compiler / Synopsys

ATPG : Tetra MAX / Synopsys

(5) To prepare the test cost information of NS-DFT, we used an in-house tool.

2-5-2 Experimental Circuits

Table 2-1 shows the characteristics of the experimental circuits without DFT. These nine circuits were used as cores. The 1st column shows the core number. The 2nd column shows the name of the core. The 3rd column shows the number of inputs. The 4th column shows the number of outputs. The 5th column denotes the number of memory elements. The 6th column shows the area in terms of gate number after logic synthesis. The 7th column shows the system clock frequency. In addition, the frequency of the scan clock is assumed to be 1/5 of the system clock of normal operation. This assumption is based on the survey of some product data. Also, the frequency of the test clock of NS-DFT is the same as the system clock at normal operation. The 8th column shows the estimated power consumption. The values are relative to the normal operation of core No.1 without DFT. The 9th column denotes the minimum number and maximum number of the scan chains, which were added to prepare test cost information of scan DFT. The values inside the parenthesis show the number of choices in DFT selection. For example, we prepared 11 cases of the test cost information about scan DFT for core No.1. Scan DFT needs 4 more inputs (clock, reset, test mode, scan enable) beside the scan chains. Therefore, the sum of the extra 4 inputs and the number of chains corresponds to the TAM width. The 10th column denotes the minimum TAM width and the maximum TAM width of NS-DFT. The values inside the parenthesis show the number of choices in DFT selection.

In this experiment, two types of DFTs are applied to each circuit: one is Scan DFT, and the other is NS-DFT. NS-DFT is a technique still under research and, for this reason, there are a few constraints when using it, such as controller and data-path must be

Table 2-1 Information of the cores

No.	name	# of Pis (bit)	# of Pos (bit)	# of FFs	Area	frequency (MHz)	power *1	Scan DFT # of chain min - max	NS-DFT TAM width min - max
1	Gcd	32	16	51	1094	100	1.00	1 - 17 (11)	16 (1)
2	Iir	20	16	192	2525	100	0.93	1 - 20 (16)	16 (1)
3	Jwf	80	80	228	4494	100	2.16	1 - 80 (29)	32 - 64 (2)
4	Lwf	32	32	83	1647	100	1.04	1 - 32 (16)	32 (1)
5	Paulin	32	32	115	4713	100	1.48	1 - 33 (18)	32 (1)
6	Risc	32	98	1284	40528	100	141.31	1 - 92 (58)	32 - 98 (3)
7	Mpeg	58	128	1942	36560	100	116.15	1 - 122 (75)	32 - 128 (3)
8	DctF	129	260	357	22650	100	19.40	1 - 179 (37)	34 - 260 (5)
9	IdctC	96	224	838	58287	100	54.77	1 - 210 (54)	32 - 224 (4)

*1 Values relative to the normal operation of core No.1

designed separately. The selected experimental circuits satisfy these constraints.

Throughout this experiment, the constraints of the test scheduling were set as follows.

(1) Maximum allowed power consumption: (Sum of each core's power consumption shown in Table 2-1) * 1.5=507.34
(2) Total area size: (Sum of each core's area size shown in Table 2-1) *1.5=258747
(3) Total TAM width: 512, 256, 128, 64

These values had to be increased in order to obtain a solution in the case of only scan DFT, and the case of only NS-DFT. For a thorough evaluation of the proposed algorithm, more experiments under suitable constraints are needed.

2-5-3 Experimental Results

We made experiments for the following three cases: (1) Scan only (number of scan chain selection), (2) NS-DFT only (TAM width selection), (3) Scan and NS-DFT (DFT method and number of scan chain, TAM width selection).

Case 1: Scan only

In case 1, each core's DFT was limited to scan design. The number of scan chains in each core is selected to reduce the total test application time. Table 2-2 shows the results of case 1. The 1st column shows the name of the SoC. The 2nd column shows each SoC's maximum TAM width. The 3rd column denotes each core's number denoted in Table 1-1. The 4th column shows each core's selected TAM width. The 5th column shows the power consumption. The power consumption results are shown relative to the normal operation figures of the original core No.1. The 6th column shows each core's area size. The 7th column shows each core's test application time. The 8th column and the 9th column shows each core's test schedule. "Start" denotes each core's test start time, and "end" denotes each core's test end time. The 10th column shows each SoC's maximum power consumption under this test schedule. The 11th column shows each SoC's total area size. The 12th column shows each SoC's total test application time, either.

In SoC1, a solution for testing all cores simultaneously was obtained. The DFT of core No.6 has the largest test application time among all cores of SoC1, even though the DFT with the smallest test application time was chosen for core No.6. Under this condition, as long as there is no other choice of DFT which improves the test application time of core No.6, the total test application time does not improve.

In SoC2 and SoC3, a solution to reduce the TAM width of the largest core No.6-9, and carries out a simultaneous test was obtained.

In SoC4, the found solution splits the test of core No.6 from the test of all other cores, i.e., core No.6 is tested independently.

Case 2: NS-DFT only

In case 2, the choice of each core's DFT was limited to NS-DFT. Table 2-3 shows the results of case 2. The meaning of each column is the same as that of Table 2-2.

We obtained a test schedule such that SoC1 and SoC2 have the same test application time.

There were two factors that contributed to obtaining this result.

First, the test length for any cores with NS-DFT does not depend on its TAM width, and initial DFT selection was not changed. Although the influence of the difference of TAM width appeared in the power consumption and the area after DFT, it did not appear on these results. Second, the test application time of core No.7 was very large, and as long as there was no choice of DFT which improves its test application time, the test application time of the whole SoC was not shortened either. In addition, in the solution for SoC3 and SoC4, DFTs with the minimum TAM width were already chosen by the initial DFT selection except for core No.3, and there was no improvement from the initial DFT selection.

Case 3: Scan and NS-DFT

In case3, each core's DFT was selected from NS-DFT or scan design. Table 2-4 shows the result of case 3. The meaning of the 1st-3rd columns is the same as that of Table 2-2 and Table 2-3. The 4th column shows the selected DFT. The meaning of the 5-13th columns is the same as that of the 4-12th columns of Table 2-2 and Table 2-3.

As for the test application time of the whole SoC, we obtained the shortest times compared to cases 1 and 2. In case 1, the test application time was longer because of the DFT selected for core No.6. However, in case 3 we could choose another DFT for core No.6 such that test application time improved. In case 2, the test application time was longer because of the DFT selected for core No.7. However, in case 3 we could choose another DFT for core No.7 such that test application time improved.

Table 2-2 Test schedule and test application time (Scan only)

Name	TAM width	No.	TAM width	Power *1	Area (gates)	TAT (10^{-6}s)	Schedule Start	Schedule End	Max Power	Total Area	Total TAT	CPU (sec)
SoC1	512	1	9	0.04	1301	50	0	50	15.40	192885	304	75.8
		2	11	0.04	3296	34	0	34				
		3	15	0.10	5409	56	0	56				
		4	11	0.05	1982	27	0	27				
		5	12	0.06	5176	43	0	43				
		6	96	6.36	45667	304	0	304				
		7	76	5.62	44331	195	0	195				
		8	55	0.82	24081	217	0	217				
		9	74	2.31	61642	244	0	244				
SoC2	256	1	9	0.04	1301	50	0	50	15.40	192885	466	55.4
		2	11	0.04	3296	34	0	34				
		3	15	0.10	5409	56	0	56				
		4	11	0.05	1982	27	0	27				
		5	12	0.06	5176	43	0	43				
		6	63	6.36	45667	466	0	466				
		7	34	5.62	44331	461	0	461				
		8	27	0.82	24081	462	0	462				
		9	74	2.31	61642	244	0	244				
SoC3	128	1	9	0.04	1301	50	0	50	15.32	192885	916	19.5
		2	11	0.04	3296	34	56	90				
		3	15	0.10	5409	56	0	56				
		4	11	0.05	1982	27	56	83				
		5	12	0.06	5176	43	0	43				
		6	34	6.36	45667	893	0	893				
		7	19	5.62	44331	916	0	916				
		8	17	0.82	24081	788	0	788				
		9	22	2.31	61642	902	0	902				
SoC4	64	1	10	0.04	1301	41	1330	1371	8.76	192885	1373	32.6
		2	11	0.04	3296	34	1330	1364				
		3	19	0.10	5409	43	1330	1373				
		4	11	0.05	1982	27	1330	1357				
		5	12	0.06	5176	43	1330	1373				
		6	63	6.36	45667	466	864	1330				
		7	25	5.62	44331	657	0	657				
		8	16	0.82	24081	843	0	843				
		9	23	2.31	61642	864	0	864				

*1 Values relative to the normal operation of core No.1 without DFT

Table 2-3 Test schedule and test application time (NS-DFT only)

Name	TAM width	No.	TAM width	Power *1	Area (gates)	TAT (10^{-6}s)	Schedule		Max Power	Total Area	Total TAT	CPU (sec)
							Start	End				
SoC1	512	1	16	1.53	1680	4	0	4	461.19	254980	835	0.6
		2	16	1.19	3229	6	0	6				
		3	64	2.92	6087	3	0	3				
		4	32	1.53	2428	3	0	3				
		5	32	1.76	5635	13	0	13				
		6	32	163.75	46966	51	0	51				
		7	32	161.06	50702	835	0	835				
		8	64	25.09	29300	15	0	15				
		9	64	102.36	108953	228	0	228				
SoC2	256	1	16	1.53	1680	4	0	4	456.74	254980	835	0.6
		2	16	1.19	3229	6	0	6				
		3	64	2.92	6087	3	228	231				
		4	32	1.53	2428	3	228	231				
		5	32	1.76	5635	13	0	13				
		6	32	163.75	46966	51	0	51				
		7	32	161.06	50702	835	0	835				
		8	64	25.09	29300	15	0	15				
		9	64	102.36	108953	228	0	228				
SoC3	128	1	16	1.53	1680	4	228	232	429.04	255172	850	0.6
		2	16	1.19	3229	6	835	841				
		3	64	2.92	6087	3	232	235				
		4	32	1.53	2428	3	228	231				
		5	32	1.76	5635	13	0	13				
		6	32	163.75	46966	51	0	51				
		7	32	161.06	50702	835	0	835				
		8	34	25.17	29396	15	835	850				
		9	32	102.46	109049	228	0	228				
SoC4	64	1	16	1.53	1680	4	228	232	263.53	255172	904	0.5
		2	16	1.19	3229	6	886	892				
		3	64	2.92	6087	3	901	904				
		4	32	1.53	2428	3	232	235				
		5	32	1.76	5635	13	835	848				
		6	32	163.75	46966	51	835	886				
		7	32	161.06	50702	835	0	835				
		8	34	25.17	29396	15	886	901				
		9	32	102.46	109049	228	0	228				

*1 Values relative to the normal operation of core No.1 without DFT

Table 2-4 Test schedule and test application time (Scan and NS-DFT)

Name	TAM width	No.	selected DFT	TAM width	Power *1	Area (gates)	TAT (10^{-6}s)	Schedule		Max Power	Total Area	Total TAT	CPU (sec)
								Start	End				
		1	NS-DFT	16	1.53	1680	4	0	4				
		2	NS-DFT	16	1.19	3229	6	0	6				
		3	NS-DFT	80	2.89	6039	3	118	121				
		4	NS-DFT	32	1.53	2428	3	118	121				
SoC1	512	5	NS-DFT	32	1.76	5635	13	0	13	200.92	201154	121	33.7
		6	NS-DFT	64	163.41	46870	51	0	51				
		7	ScanDFT	126	5.62	44331	118	0	118				
		8	NS-DFT	64	25.09	29300	15	0	15				
		9	ScanDFT	172	2.31	61642	112	0	112				
		1	NS-DFT	16	1.53	1680	4	169	173				
		2	NS-DFT	16	1.19	3229	6	169	175				
		3	NS-DFT	80	2.89	6039	3	169	172				
		4	NS-DFT	32	1.53	2428	3	169	172				
SoC2	256	5	NS-DFT	32	1.76	5635	13	169	182	171.69	201250	184	55.1
		6	NS-DFT	32	163.75	46966	51	0	51				
		7	ScanDFT	112	5.62	44331	132	0	132				
		8	NS-DFT	64	25.09	29300	15	169	184				
		9	ScanDFT	109	2.31	61642	169	0	169				
		1	NS-DFT	16	1.53	1680	4	237	241				
		2	NS-DFT	16	1.19	3229	6	237	243				
		3	NS-DFT	80	2.89	6039	3	252	255				
		4	NS-DFT	32	1.53	2428	3	252	255				
SoC3	128	5	NS-DFT	32	1.76	5635	13	237	250	271.84	248657	255	26.4
		6	NS-DFT	32	163.75	46966	51	0	51				
		7	ScanDFT	63	5.62	44331	237	0	237				
		8	NS-DFT	64	25.09	29300	15	237	252				
		9	NS-DFT	32	102.46	109049	228	0	228				
		1	NS-DFT	16	1.53	1680	4	480	484				
		2	NS-DFT	16	1.19	3229	6	480	486				
		3	NS-DFT	64	2.92	6087	3	493	496				
		4	NS-DFT	32	1.53	2428	3	496	499				
SoC4	64	5	NS-DFT	32	1.76	5635	13	480	493	266.21	248705	499	28.5
		6	NS-DFT	32	163.75	46966	51	237	288				
		7	ScanDFT	63	5.62	44331	237	0	237				
		8	NS-DFT	64	25.09	29300	15	465	480				
		9	NS-DFT	32	102.46	109049	228	237	465				

*1 Values relative to the normal operation of core No.1 without DFT

Table 2-5 Comparison of test application time

Name	TAM width	case1 [A]	case2 [B]	case3 [C]	C/A	C/B
SoC1	512	304	835	121	0.40	0.14
SoC2	256	466	835	184	0.39	0.22
SoC3	128	916	850	255	0.28	0.30
SoC4	64	1373	904	499	0.36	0.55

Table 2-5 shows the comparison of test application time. The 1st column shows SoC name. The 2nd column shows total TAM width. The 3rd-5th columns show the total test application time of cases 1-3, respectively. The 6th column shows the ratio of test application time in case 3 to the test application time in case 1. The 7th column shows the ratio of test application time in case 3 to the test application time in case 2. In case 3, the total test application times were shortened from 60% to 72% compared with case 1, and from 45% to 86% compared with case 2.

Figure 2-7 shows the test schedule of SoC4. The vertical axis shows the TAM width and the horizontal axis shows the test application time. In case 2, of all cores except for Mpeg, the test application times are shorter than that of care1. So, the Mpeg is the bottleneck of case2. In case3, Scan DFT is selected as the DFT of Mpeg, and the total test application time is drastically improved.

These experimental results show that the differences in the selection scope of DFT drastically changes the test cost of an SoC. Therefore, the usefulness of having a database with the test cost information of several DFTs, and using this database to optimize DFT selection to reduce test cost in early stages of design flow was shown.

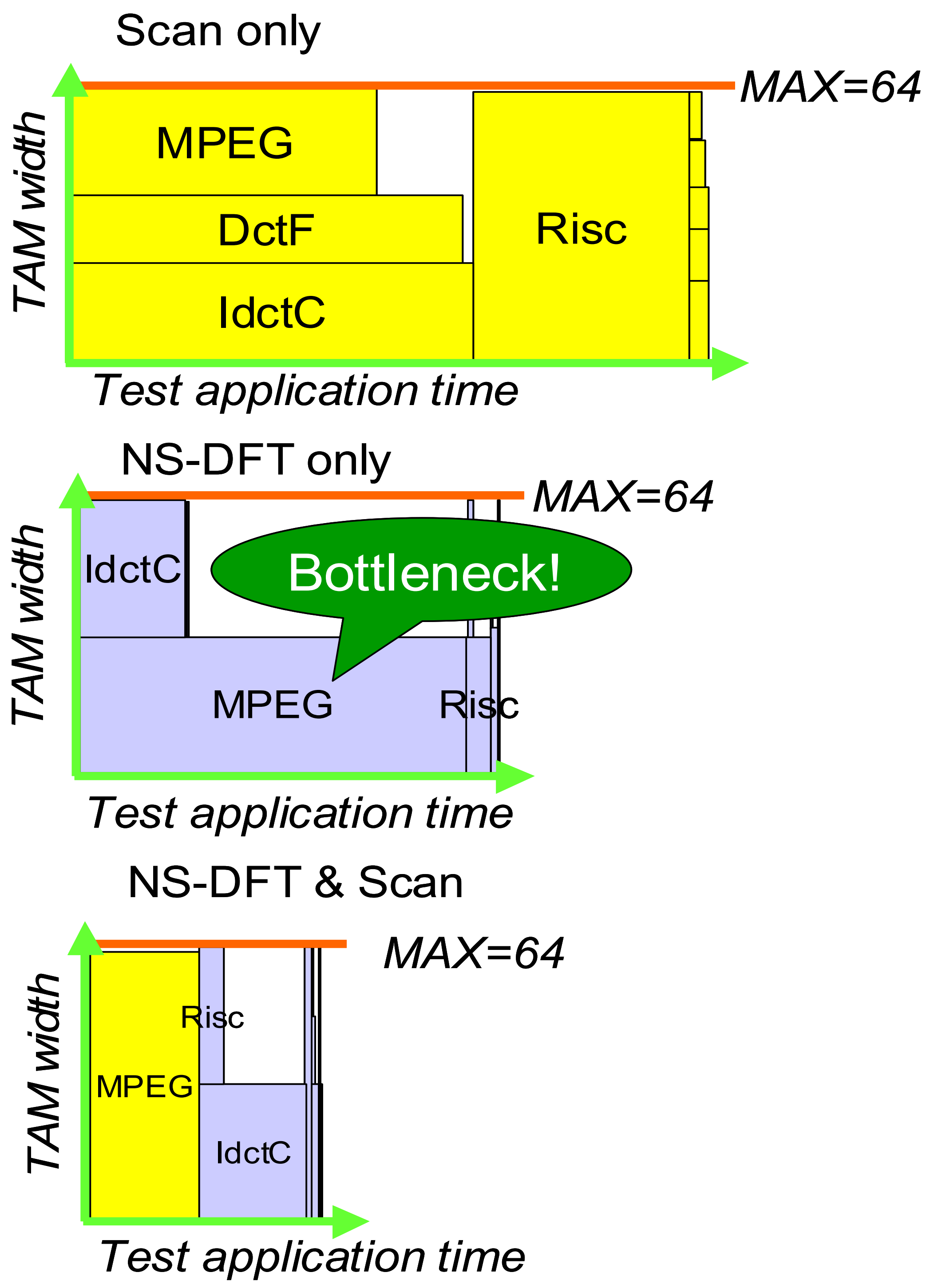

Fig. 2-7 Test schedule of SoC4

Table 2-6 shows the test application time, which is calculated in cases 1 to 3, and the lower bounds of optimum solution, which is denoted in section 4. The 1st column shows the case number. The 2nd column shows the name of the SoC. The 3rd column shows the total TAM width. The 4th column shows total test application time, which is calculated in this experiment. The 5th column shows the lower bound of test application time, which is described in section 4. The 6th column shows the ratio of experimental results relative to the lower bounds. The last column shows the CPU time of the experiments.

In cases 1, 2, and 3, the numbers of combinations of DFT selection are $1.28\text{x}10^{13}$, 360, $2.37\text{x}10^{13}$, respectively. In the worst case, the proposed algorithm performs exhaustive search, and the execution time is possible to be so large that the experiment cannot be completed. However, in this experiment, the execution time is less than 100 seconds. Although the algorithm may have ended without carrying out sufficient search, the ratio of experimental results to the lower bounds shows that the experimental results are less than 1.6 times larger than the optimum solutions.

Table 2-6 Test application time and lower bound of optimum solution

	Name	TAM width	Total TAT	LB	TAT/LB	CPU (sec)
case1	SoC1	512	304	304	1.00	75.8
	SoC2	256	466	304	1.53	55.4
	SoC3	128	916	596	1.54	19.5
	SoC4	64	1373	1191	1.15	32.6
case2	SoC1	512	835	835	1.00	0.6
	SoC2	256	835	835	1.00	0.6
	SoC3	128	850	835	1.02	0.6
	SoC4	64	904	835	1.08	0.5
case3	SoC1	512	121	118	1.03	33.7
	SoC2	256	184	118	1.56	55.1
	SoC3	128	255	198	1.29	26.4
	SoC4	64	499	396	1.26	28.5

2-6 Conclusions

The framework of an SoC test architecture generation was proposed. The framework contains a database, which stores the test cost information on several DFTs for every core, and DFT selection part. In the framework, each core's DFT is selected for reducing the test application time using test cost information database in the early phase of the design flow. Moreover, the DFT selection problem was formulated and the algorithm, which solves this was proposed. Experimental results show that bottlenecks in test application time when using the single DFT method for all cores in a SoC is reduced by performing DFT selection from two types of DFTs. As a result, the whole test application time is shortened.

3 A MEMORY GROUPING METHOD FOR REDUCING MEMORY BIST LOGIC OF SYSTEM-ON-CHIPS

3-1 Introduction

With the increasing number of functions being included in SoCs, many memories with different sizes and frequencies are being used. Recently, SoCs contain hundreds of memories. Testing all the memories in these SoCs sequentially would take a long time. Therefore, a memory BIST design that allows two or more memories to be tested simultaneously is needed. However, due to power-consumption constraints, not all memories can be activated at the same time. To solve this problem, a scheduling technique for minimizing the test application time under power-consumption constraints is needed. Adding individual circuits for memory BISTs to lots of small memories would result in huge area overheads. To reduce these overheads, memory BIST logic must be able to be shared.

A BIST architecture, based on a single micro-programmable BIST processor and a set of memory wrappers, was proposed to simplify the testing of systems containing many distributed SRAMs of different sizes [20]. To reduce the BIST area overhead, it was proposed to share a single wrapper between a cluster of SRAMs (same type, width, and addressing space). There is another architecture that can connect memories that have different widths or addressing spaces and share BIST logic.[21]. In the architecture, single memory BIST logic can test any number of memories. Memories can be tested serially or in parallel. Each memory to test is assigned to a particular step. All memories in step 1 are tested before memories in step 2, and so on. The designer can select a satisfactory assign of memories in consideration of the test time, the diagnostic resolution and the overhead. But there is no deterministic algorithm to find an optimal assign of memories.

In this chapter, two types of memory-connection methods for BIST wrapper sharing are proposed. A memory-grouping problem for test circuit minimization under constraints of power consumption and test application time is also formulated together with an algorithm that solves the problem. In addition, the effectiveness of this technique is demonstrated experimentally. This chapter is organized as follows. In section 3-2, memory BIST logic sharing method is described. In section 3-3, the memory-grouping problem and an algorithm to solve the problem are presented. The experimental results are shown in section 3-4. Finally, section 3-5 concludes this chapter.

3-2 Memory BIST Logic Sharing

In this section, two methods of BIST logic sharing for single port and word access memory are described. Figure 3-1 shows an example of a memory BIST wrapper. The data generator generates input test sequences. The address generator generates read and write addresses and the response analyzer captures test output responses and detects faults. The by-pass FFs are not used to test memory, but are used to handle the memory interface signal during a scan test. The area of the address generator, data generator, and response analyzer are almost proportional to the bit width of the address, input data, and output data, respectively. However, some of these logics can be shared by different memories wherever the number of words or the data bit width are the same; hence, the area of test circuits can be reduced. In this thesis, the following two memory connection methods for memory BIST logic sharing are treated: **parallel connection** and **serial connection**.

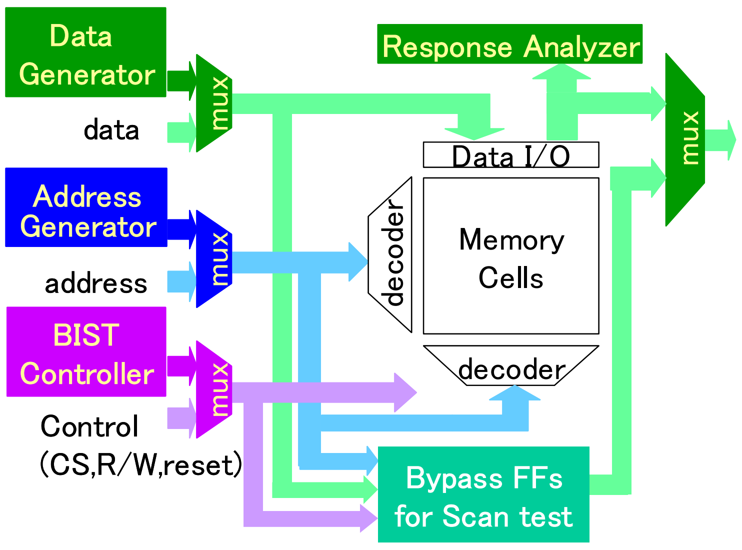

Fig. 3-1 Memory BIST wrapper

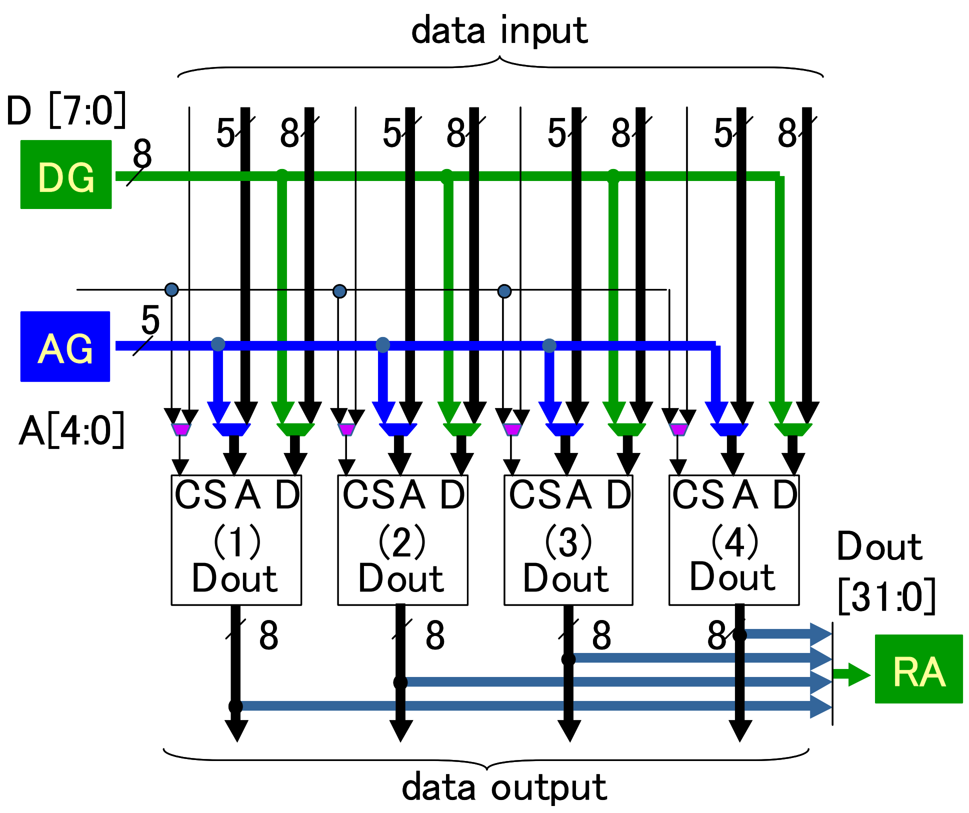

Fig. 3-2 Parallel connection of memories

Parallel connection can be used to connect memories that have the same number of words. Figure 3-2 shows an example of parallel connection.

In this example, three data and address generators are reduced to one by distributing the same test data and address signals from a couple of data and address generators to (1) - (4), enabling four memories to be tested simultaneously.

Serial connection allows memories with the same bit width to be connected. Figure 3-3 shows an example of four serially connected 8x32 word memories. In this example, the four memories are tested as an 8x128 word memory. The address generator generates an additional 2bit signal, and the signal is used to select the memories from (1) - (4), enabling the four memories to be tested serially. If all the memories have individual BIST logic, a 32-bit data generator and response analyzer are required, but in this example, all the memories can be tested using a shared 8bit generator and 8bit response analyzer.

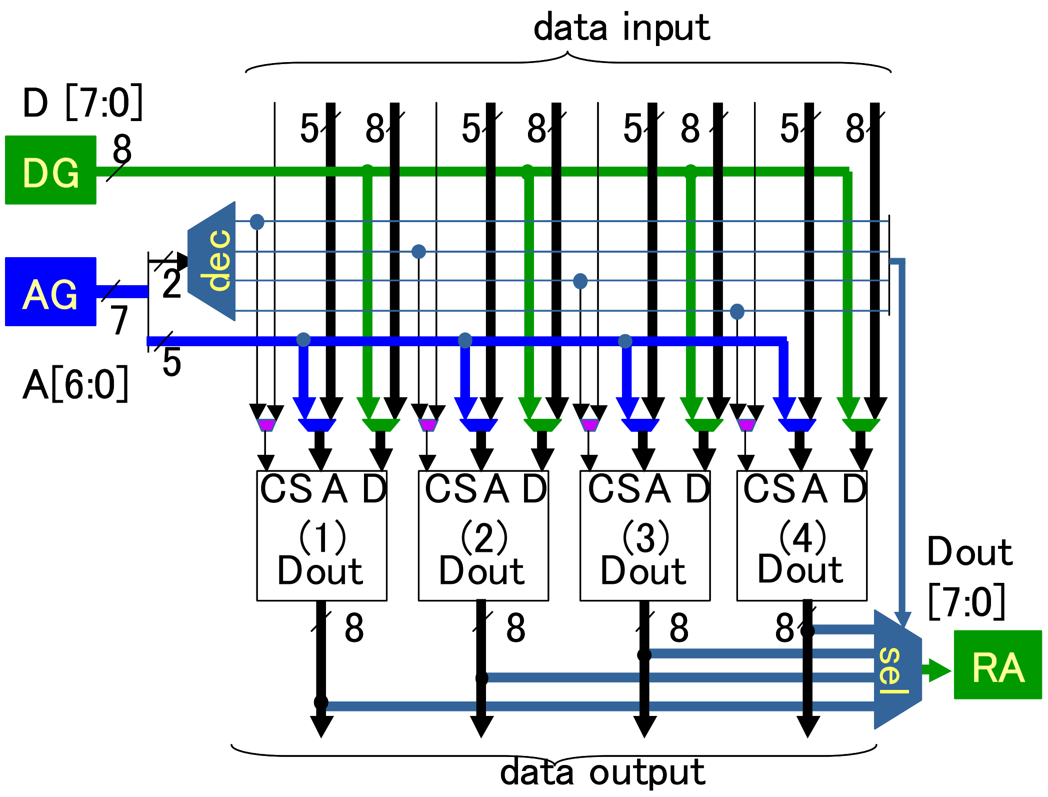

Fig. 3-3 Serial connection of memories

Serial connection reduces the area more than parallel connection and also uses less power than parallel connection. However, the time required for serial connection testing is longer than that for parallel connection testing.

To achieve the minimum area and a reasonable test application time under power consumption constraints, the type of memory connection should be considered during decisions on memory grouping. The layout design must also take into account distance constraints in relation to these connections.

3-3 Memory-Grouping Problem and Algorithm

3-3-1 Formulation of Memory-Grouping Problem

In this subsection, a memory-grouping problem is formulated. It is assumed that the following information for each memory m_i is given:

- b_i: data bit width of m_i
- w_i: word depth of m_i
- p_i: maximum power consumption of testing m_i
- f_i: operating frequency of m_i
- x_i: X coordinate of m_i, y_i: Y coordinate of m_i

We define two types of compatibility, namely p-compatibility and s-compatibility, as follows:

Given a set of memories $V=\{m_1, m_2, \ldots m_n\}$, a pair of memories $m_i, m_j \in V$ is ***p-compatible*** if they satisfy the following conditions:

$$w_i = w_j \tag{1}$$

$$f_i = f_j \tag{2}$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < D \tag{3}$$

D is a constraint value that the designer decides according to the design condition.

P-compatibility is represented by a graph $G_p = (V, E_p)$, where V is a set of a memory and the edge between a pair of vertices $(m_i, m_j). \in E_p$ exists if m_i and m_j are ***p-compatible***. If a set of memories can be connected in parallel, the graph induced on G_p by the memories has to be a clique.

In the same way, a pair of memories $m_i, m_j \in V$ is ***s-compatible*** if they satisfy the following conditions:

$$b_i = b_j \tag{4}$$

$$f_i = f_j \tag{5}$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < D \tag{6}$$

S-compatibility is represented by a graph $G_s = (V, E_s)$, where V is a set of memories and the edge between a pair of vertices $(m_i, m_j). \in E_s$ exists if m_i and m_j are ***s-compatible***. If a set of memories can be connected serially, the graph induced on G_s by the memories has to be a clique.

To design memory BIST wrappers using these techniques for memory BIST logic sharing, we have to find a partition of V such that the memories that share the wrapper are included in the same block. Moreover, the partition $\pi = \{B_1, B_2, \ldots B_n\}$ has to satisfy the following conditions:

G_{ip} is the graph induced on G_p by block B_i.

G_{is} is the graph induced on G_s by block B_i.

G_{ip} or G_{is} is a clique.

When only the graph G_{ip} (G_{is}) is a clique, the memories included in B_i are connected in parallel (serially). Figure 3-4 shows an example of a compatibility graph and our target partition. For a given set of memories M1-5, p-compatibility, and s-compatibility graphs under the distance constraint D=30 can be generated as shown in Figure 3-4 (a) and (b), respectively. Figure 3-4 (c) shows an example of our target partition. The partition has two blocks, B_1 and B_2. B_1 and B_2 are the node set of the clique of the s-compatibility and p-compatibility graphs, respectively.

	bits	words	frequency (MHz)	location	
				x	y
M1	32	128	133	20	10
M2	32	128	133	40	10
M3	32	128	133	41	10
M4	16	128	133	52	10
M5	16	128	133	53	10

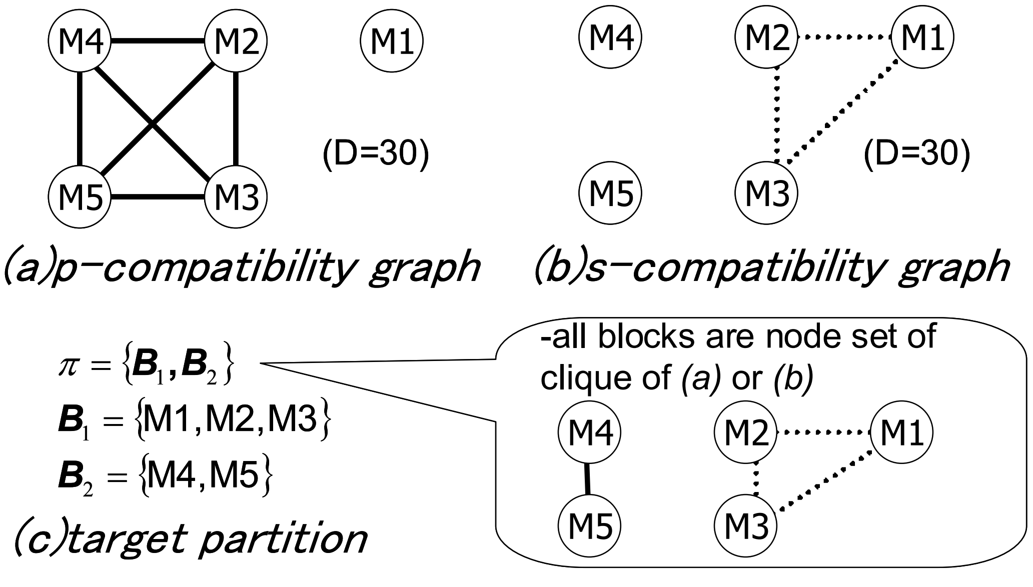

Fig. 3-4 Compatibility graphs and target partition

For a partition π, we can calculate the area of the BIST wrapper, test application time, and power consumption of each block. The area and test application time depend on the test-pattern algorithm. In this work, these were calculated according to a published design [24] using an 8N algorithm as follows.

If the connection type of block $B_i=\{m_1, m_2, \ldots m_k\}$ is a parallel connection,

$$\text{Area } S_{Bi} = 0.75(\log_2(w_{Bi}))^2 + 2k\log_2(w_{Bi}) + 18\sum_{l=1}^{k} b_l + 25\log_2(w_{Bi}) + 3\max_l(b_l) + 66 \quad (7)$$

$$\text{Power consumption } P_{Bi}=\sum_{l=1}^{k} p_l \quad (8)$$

$$\text{Test application time } T_{Bi}=8\times w_{Bi} / f_{Bi} \quad (9)$$

$(f_{Bi}=f_1=f_2=\ldots=f_k)$

If the connection type of block $B_j=\{m_1, m_2, \ldots m_k\}$ is a serial connection,

Area S_{Bj} =

$$0.75\left(\log_2\left(\sum_{l=1}^{k} w_l\right)\right)^2 + 2k\log_2\left(\sum_{l=1}^{k} w_l\right) + 25\log_2\left(\sum_{l=1}^{k} w_l\right) + k(\log_2 k) + 9b_{Bj}k + 14b_{Bj} + 8k + 61 \quad (10)$$

$(b_{Bj}=b_1=b_2=\ldots=b_k)$

$$\text{Power consumption } P_{Bj}=\max_l(p_l) \quad (11)$$

Test application time

$$T_{Bj}=8\times\left(\sum_{l=1}^{k} w_l\right)/ f_{Bj}\times(\text{number of background patterns}) \quad (12)$$

The expressions for area calculation (7) and (10) do not consider the influence of timing conditions, but feedback is available from previous designs.

Parallel-connected memories are tested concurrently, and the power consumption is the sum of the power consumption of each memory. In contrast, serial-connected memories are activated one by one. Therefore, the power consumption is the maximum power consumption of the connected memories.

When a partition π as described in 3.1 is found, the area, power consumption, and test application time of each block are calculated using the above expression.

The total area of the memory BIST wrappers S_{total} is calculated as the sum of S_{Bi}.

$$S_{total} = \sum_{i=1}^{n} S_{Bi} \tag{13}$$

To control each memory BIST wrapper, at least one BIST controller must be used. In this study, the number of memory BIST wrappers was reduced by using the proposed connections. There was therefore no increase in the number of controllers. In addition, our target design includes a lot of memories so that the area of the memory BIST wrappers is predominant. Therefore the area of the BIST controllers is disregarded. But if there is a large difference in BIST controllers between parallel and serial connection, S_{total} should include the area of BIST controllers. The difference in the BIST controller area will depend on the BIST architecture and algorithm used.

If there are many memories close to each other, the wiring congestion may also need to be taken into consideration. To add the parameter that reflects the amount of wiring to the area calculation is our future work.

To calculate the total test application time of a memory BIST under a power-consumption constraint, we used a rectangle packing algorithm that has been described elsewhere [14]. The algorithm optimizes the test schedule of each core so that the total test application time of an SoC is minimized under maximum power constraints. The inputs of the scheduling algorithm are the maximum allowed power consumption, the test application time, and the power consumption of each core. In this study, we considered a block to be a core. Therefore, we input $\{P_{Bj}\}$ $\{T_{Bj}\}$ as the information for each core. In addition, we assumed the bit width of the inter-connect between each wrapper and control logic remained unchanged. We therefore disregarded the maximum TAM width.

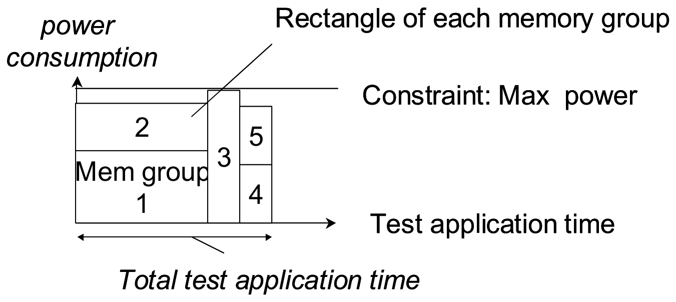

Fig. 3-5 Test scheduling using rectangle packing

In this work, rectangles represent the test application time and power consumption of each memory group were packed within limits representing the power consumption as shown in Figure 3-5. The packing within the limits is determined so that the total test application time is as short as possible. The left end of each rectangle shows the test start time of the corresponding memory group.

To reduce the total area of memory BIST wrappers by memory BIST logic sharing, we formulated the following memory-grouping problem.

Inputs:

a) A set of memories S and information for each memory: $M=M_i$ (b_i, w_i, p_i, f_i, x_i y_i) where, b_i, w_i, p_i, f_i, x_i, and y_i are as follows:

b_i: data bit width of m_i

w_i: word depth of m_i

p_i: maximum power consumption of testing m_i

f_i: frequency of m_i

x_i: X coordinate of m_i

y_i: Y coordinate of m_i

Outputs:

a) A partition π of a given set of memories S for which all the blocks satisfy the following conditions:

G_{ip} is the graph induced on G_p by block B_i.

G_{is} is the graph induced on G_s by block B_i.

G_{ip} or G_{is} is a clique.

b) Type of connection of each block

c) Test schedule of each memory

Constraints:

a) Maximum distance of memory connection: D

b) Maximum available peak power of the SoC: P

c) Maximum test application time of memory: T

Objective:

To minimize S_{total}.

To solve this problem, an algorithm is proposed below.

3-3-2 Memory-Grouping Algorithm

Figure 3-6 shows the memory-grouping algorithm. In step 1, the algorithm creates an s-compatibility graph. In step 2, the minimum cut edge is calculated and deleted from the s-compatibility graph. As a result of this operation, the graph is divided, leaving a high possibility of a reduction in area. In step 3, if the graph is not divided as a clique partition, the algorithm returns to step 2. In step 4, the algorithm calculates the test schedule. In step 5, if the test scheduling fails, the algorithm returns to step 2 and divides the graph. If all the memories are divided individually and the scheduling fails, it means that there is no solution under the given constraints. In step 6, the algorithm gathers blocks that have only one memory into one block, and searches for the partition at which S_{total} is minimized using p-compatibility. In this second search, it does not consider blocks that are determined to include two or more memories by the first search. These are considered to be suitable for serial connection, while the rest are considered to be suitable for parallel connection.

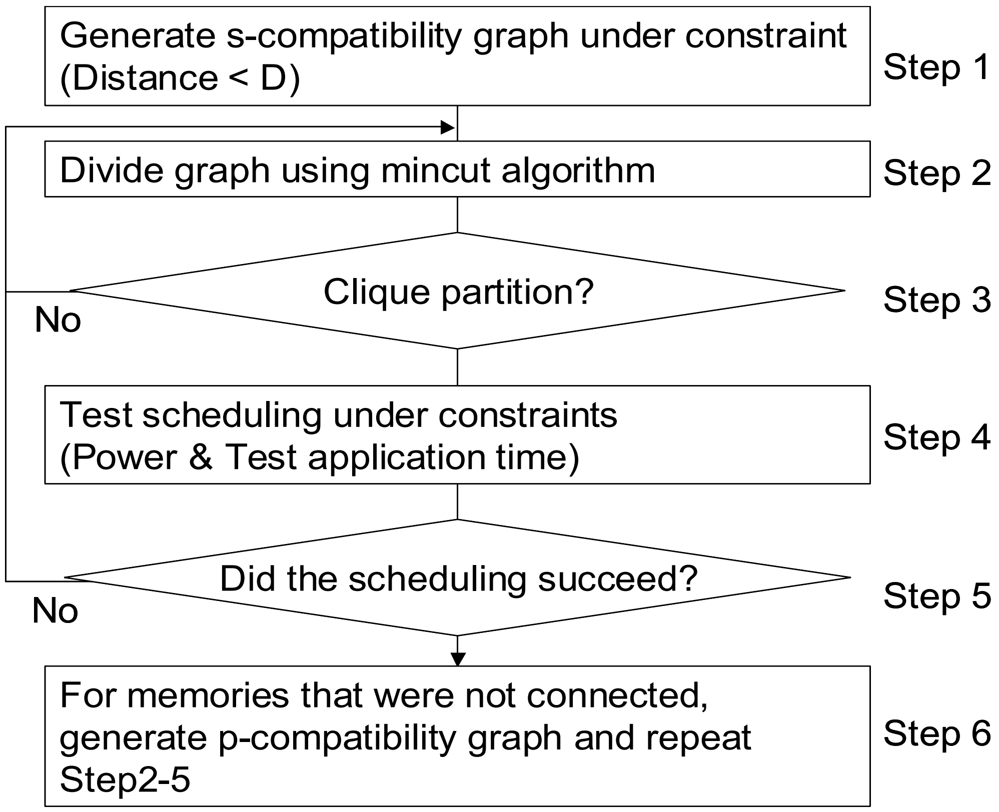
Generate s-compatibility graph under constraint (Distance < D)
Step 1
Divide graph using mincut algorithm
Step 2
Clique partition?
No
Step 3
Test scheduling under constraints (Power & Test application time)
Step 4
Did the scheduling succeed?
No
Step 5
For memories that were not connected, generate p-compatibility graph and repeat Step2-5
Step 6

Fig. 3-6 Memory grouping algorithm

Our proposed algorithm repeats the division process from a 0-partition, that is, only one block that includes all the memories, to obtain the target partition. As the algorithm divides the block, S_{total} increases. To reduce S_{total}, we use the following heuristics. As shown in Figure 3-7, we introduce the weight of an edge that represents the sum of the reduced bit with the data generator and response analyzers resulting from the connection. M1-M5 are the same set of memories that were denoted in Figure 3-4. For example, M1 and M2 have 32-bit data inputs and outputs. If these memories are connected using serial connection, we can reduce the 32-bit data generator and 32-bit response analyzer. So the weight of the edge {M1, M2} in the s-compatibility graph is calculated as 32+32=64.

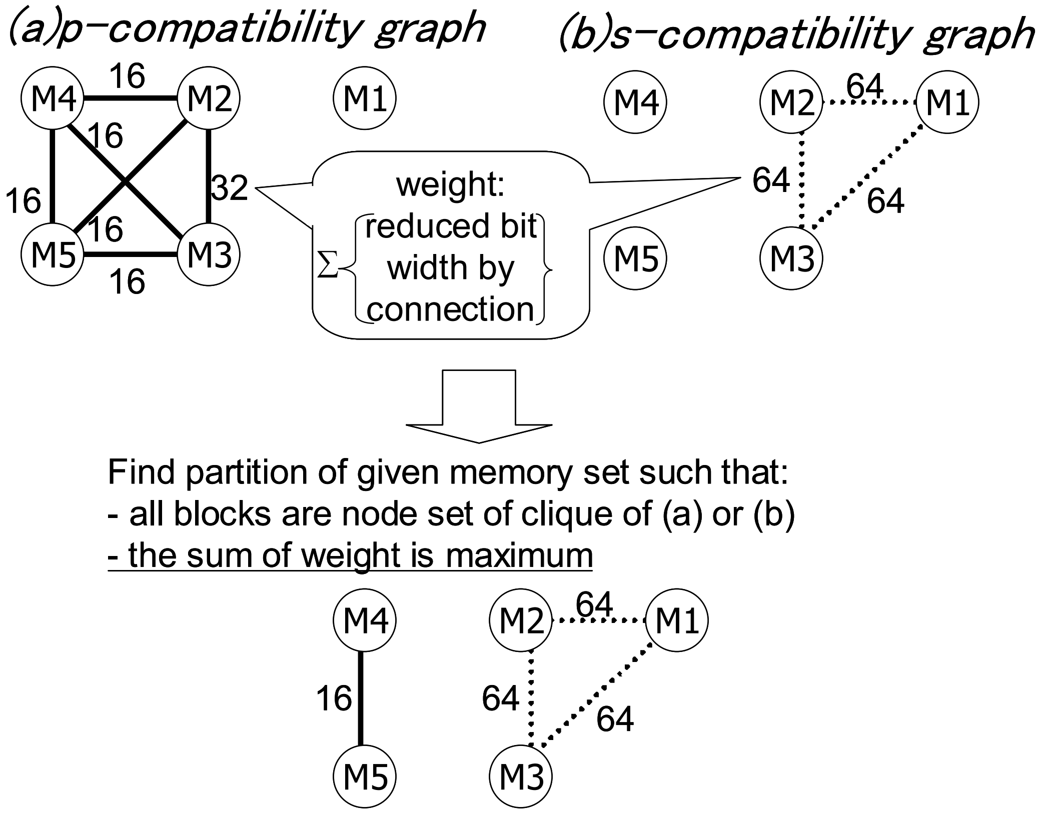
(a)p-compatibility graph
(b)s-compatibility graph
M4
M2
M1
M5
M3
16
32
64
weight:
Σ reduced bit width by connection
Find partition of given memory set such that:
- all blocks are node set of clique of (a) or (b)
- the sum of weight is maximum

Fig. 3-7 Heuristic of graph division

To ensure that the area is reduced as much as possible, we use a min-cut method [22][23] in the graph division process. The following strategies are also used to decide the compatibility of each block of the partition. Serial connection reduces the area more than parallel connection, and it also consumes less power. Therefore, it is possible that giving priority to serial connection reduces S_{total}. Based on this prospect, the proposed algorithm searches for the partition that minimizes S_{total} using only s-compatibility in the first search. Figure 3-8 shows the pseudo code of the Memory Grouping Algorithm.

First, the algorithm initializes variables. The minimum value of S_{total} is stored into S_{min}, and, in the first step, S_{min} is set to the total area of memory BIST wrapper without sharing. The partition of a set of memory S is stored into π, and the initial partition is set to 0-partition of S. (line 1-2).

Next, the algorithm creates two compatibility graphs (line 3), and select s-compatibility graph as the graph G that is used to find partition (line 4).

In order to check the compatibility of each block, the algorithm construct a set of graph C_{all} (line 6). Each graph G_i that is the member of C_{all} is induced on G by block B_i that is the member of π.

Then, for all B_i that include two or more memories, execute the following operations (line 7-21).

The minimum cut edge is calculated and delete them from G_i. By this operation, the vertex set B_i is divided into two blocks, leaving much possibility of the area reduction. If all the graph of new graph set C_{all} are clique, calculate S_{total} and test schedule of the new partition π_{tmp}. If $S_{min}>S_{total}$ and the test scheduling succeeded, π_{tmp} is stored into π_{best} as the best partition, and S_{total} is stored into S_{min} (line 8-17). If there is a graph G_i that is not a clique, or the test scheduling failed, π_{tmp} is stored into π_{next} (line 18-20).

If there is no partition that should be tried, the first search is end (line22-24). Then the algorithm stores p-compatibility graph into G, and collects the blocks that have only one memory into one block (line25-28). Then, the algorithm searches for the partition that S_{total} is minimized using p-compatibility (line5-24).

This algorithm performs n(n-1) times division and scheduling in the worst case. The complexity of the scheduling algorithm and min-cut algorithm are $O(V\log V)$ and $O(V^2\log V)$, respectively. Therefore the complexity of this algorithm is $O(V^3\log V)$.

```
Procedure Memory_Grouping (M, P, T, D){
1   $S_{min}$= the total area of memory BIST wrapper without sharing; maxedgenum=0; edgenum=0;
2   $\pi=\{B\}$, $B=\{m_1, m_2, \ldots m_n\}$; $\pi_{tmp}=\phi$; $\pi_{next}=\phi$; $\pi_{best}=\phi$; $\pi_{s-compatible}=\phi$;
3   $G_s$ = s-compatibility_graph of B; $G_p$ = p-compatibility_graph of B;
4   $G = G_s$;
5   loop:
6   Construct a set of graph $C_{all}=\{ G_i \mid G_i$ is induced graph on G by $B_i \in \pi \}$
7   for( $\{B_i \in (\pi - \pi_{s-compatible}) \mid$ which includes two or more memories$\}$){
8       delete min-cut edge from $G_i$, make a set of graph $C_{min}=\{G_{i1}, G_{i2}\}$;
9       $C_{all}= (C_{all}- G_i) \cup C_{min}$;
10      Set edgenum=$\sum_j$(the number of edges of $G_j \in C_{all}$);
11      Set a partition $\pi_{tmp}=\{B_j \mid$ vertex set of $G_j \in C_{all}\}$;if all $G_j$ are clique,calculate $S_{total}$ of $\pi_{tmp}$
12      if(($\forall G_j \in C_{all}$, $G_j$ is clique ) $\wedge$ ($S_{min}>S_{total}$ of $\pi_{tmp}$)){
13          calculate $T_{total}$=Schedule(P, $\{P_{Bj}\}$, $\{T_{Bj}\}$);
14          if((Schedule succeeded) $\wedge$ ($T_{total} \le T$)){
15              $S_{min} = S_{total}$; $\pi_{best}=\pi_{tmp}$;
16          }
17      }
18      if(edgenum > maxedgenum $\wedge$ ((Schedule failed, or $T_{total} \le T$) $\vee$
          ($\exists G_j \in C_{all}$, $G_j$ is not a clique))){
19        $\pi_{next}=\pi_{tmp}$; maxedgenum=edgenum;
20      }
21  }
22  if($\pi_{next} \ne \phi$){
23      $\pi = \pi_{next}$; $\pi_{next}=\phi$; go to loop;
24  }
25  else if($G=G_s$){
26      $G = G_p$; $\pi_{s-compatible}=\{B_j \in \pi_{best} \mid$ which includes two or more memories $\}$;
27      $B_s=\bigcup_k B_k$   ($B_k \in (\pi_{best} - \pi_{s-compatible})$); $\pi=\{B_s\} \cup \pi_{s-compatible}$; go to loop;
28  }else{end;}
```

Fig. 3-8 Memory grouping algorithm (pseudo code)

3-4 Experimental Results

We carried out experiments to evaluate the proposed method. The proposed algorithm was implemented in C and the experiments were conducted on a 600-MHz Windows PC. Table 3-1 shows the information in each memory used in the experiment. The 2-4th columns denote the data bit width, word depth, and operating frequencies, respectively. The 5th column shows the power consumption. In this experiment, the power consumption of each memory was a relative value in which memory No. 1 was assumed to be 100 under the following assumption:

(1) The area is proportional to (number of words × number of bits).

(2) The power consumption is proportional to the area.

(3) The power consumption is proportional to the frequency.

The 6th and 7th columns show location. In this experiment, the number of memories was varied between 3 and 50, and the program was executed respectively. When the number of memories was N<11, we used No. 1 to N, and for the rest, we extended the same set of No.1-10, with the Y coordinate changing between 20 to 50.

Table 3-1 Algorithm input information on memories

No.	# data bit width	#Words	Frequency (MHz)	Power *1	Location	
					X	Y
1	16	128	133	100	10	10, 20, 30, 40, 50
2	16	128	133	100	20	
3	16	128	266	200	30	
4	16	128	266	200	40	
5	16	256	133	200	50	
6	16	256	133	200	60	
7	16	256	133	200	70	
8	16	256	133	200	80	
9	32	512	133	400	90	
10	32	512	133	400	100	

*1 Relative values in which memory No.1 is assumed to be 100

In an actual test, several background patterns (e.g. marching, checker, checker-bar) are used, but in this experiment, the test application time was calculated by assuming the number of background patterns=1. In addition, the following constraint values were used:

Maximum distance of memory connection: D=40

Maximum available peak power of the SoC: P=5000

Maximum test application time of memory: T=300 μs

Experiments were carried out for the following five cases:

(1) Not shared (all the memories had individual BIST wrappers)

(2) Parallel connection (memory BIST logic was shared using only parallel connection as described in the proposed technique)

(3) Serial connection (memory BIST logic was shared using only serial connection as described in the proposed technique)

(4) Parallel and serial connection (memory BIST logic was shared using both parallel and serial connection as described in the proposed technique)

(5) Exhaustive search (memory BIST logic was shared using only parallel connection after an exhaustive search).

Table 3-2 shows the experimental results. The first column shows the number of memories and the second column shows the total area of memory BIST wrappers without sharing. Columns 3-5 show the total area of memory BIST wrappers using the proposed techniques. The third column shows the results of using only parallel connection, while the fourth column shows the results of using only serial connection. The fifth column shows the results of using both parallel and serial connection and the sixth column shows the minimum solution obtained using an exhaustive search. The last column shows the ratio of the results of S&P relative to the minimum solutions.

We were only able to complete an exhaustive search when the number of memories was less than 7. In these cases, the results of the exhaustive search showed that the memory BIST logic sharing technique reduced the area of the BIST wrappers by between 21.59 and 47.83% as minimum solutions. When the number of memories is less than 6, proposed technique achieved minimum solution. When the number of memories are 6 and 7, the results are 12% and 33% large than minimum solutions, respectively. The quality of the solution seems to deteriorate as the size of the problem grows. There may be room for improving the quality of solution.

The average reduction ratio for parallel connection, serial connection, and parallel and serial connection were 21.08%, 37.25%, and 40.55%, respectively.

Table 3-2 Area of memory-BIST logic

#of mem	not shared	Proposed algorithm			exhaustive	S&P/ exhaustive
		P only	S only	S&P		
3	2289	1967	1660	1660	1660	1.00
4	2913	2591	2284	2284	2284	1.00
5	3537	2893	2279	2279	2279	1.00
6	4203	3559	3044	2722	2415	1.13
7	4869	3863	3690	3368	2540	1.33
8	5535	4529	3719	3397	N/A	N/A
9	6201	4793	3828	3506		
10	7242	5427	3122	3122		
11	8283	6021	6447	5678		
12	8907	7539	4703	4703		
13	9531	7394	5411	5089		
14	10155	7696	5406	5406		
15	10779	7998	5401	5401		
20	14484	10854	6769	6769		
30	21726	16281	10455	10455		
40	28968	22070	23784	19662		
50	36210	27497	22964	21551		

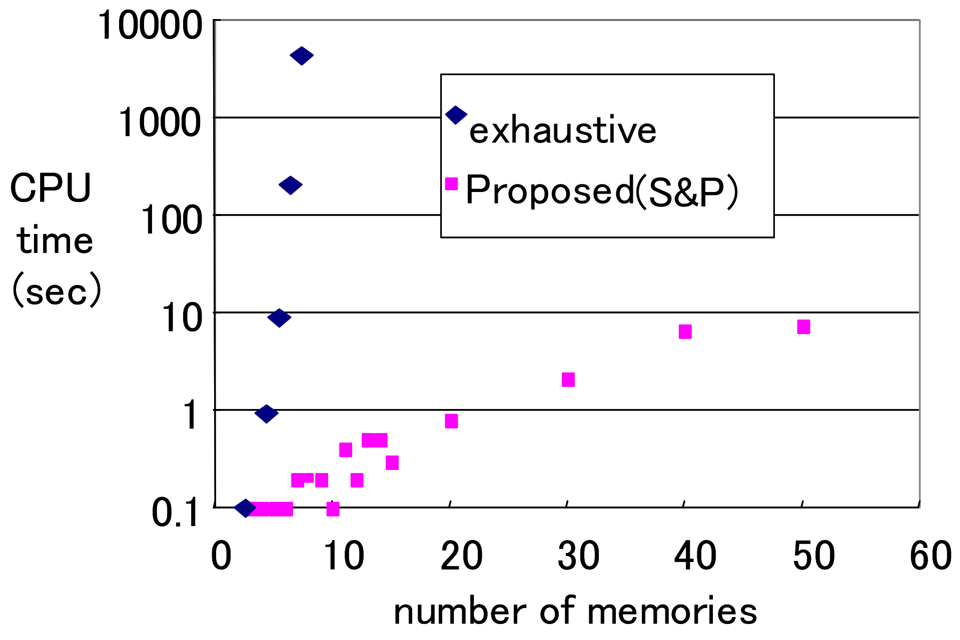

Fig. 3-9 CPU time for memory grouping program

In all cases, parallel and serial connection achieved the best solution. This result demonstrates that selection from two types of connection methods reduces the area more than using a single connection method.

Finally, Figure 3-9 shows the execution time of the implemented memory-grouping program. In all cases, the program was executed within 10 seconds using the proposed algorithm. The technique thus obtained good results within a very short CPU time so it is suitable for practical application.

3-5 Conclusion

A memory grouping problem was formulated and an algorithm to solve the problem was proposed. Experimental results showed that the proposed method reduced the area of memory BIST wrappers by up to 40.55%. It was also shown that the ability to select from two types of connection methods reduced the area more than using a single connection method.

4 CONCLUSION AND FUTURE WORK

4-1 Summary of the Thesis

As the increasing of functionality of SoCs, SoCs are getting large, and the more test application time would be needed. Furthermore, the functionality of SoCs increasing causes high overhead of memory BIST logic. This thesis presents the methods for easing these problems.

Chapter 2 presented a framework of an SoC test architecture generation. The framework contains a database, which stores the test cost information on several DFTs for every core, and DFT selection part. In the framework, each core’s DFT is selected for reducing the test application time using test cost information database in the early phase of the design flow. Moreover, the DFT selection problem was formulated and the algorithm to solve the problem was proposed. Experimental results show that bottlenecks in test application time when using the single DFT method for all cores in a SoC is reduced by performing DFT selection from two types of DFTs. As a result, the whole test application time is shortened.

Chapter 3 presented a memory grouping problem formulation and an algorithm to solve the problem. Experimental results showed that the proposed method reduced the area of memory BIST wrappers by up to 40.55%. It was also shown that the ability to select from two types of connection methods reduced the area more than using a single connection method.

Finally, I show the design phase in which the memory grouping should be done. Figure 4-1 shows the new framework of SoC test architecture generation that presented in chapter 2. Generally, memory BIST logic is often tested by scan test. In such case, it is necessary to include the test of memory BIST logic in the test schedule of the DFT selection. Therefore, it is preferable to build in memory grouping and memory BIST logic before the selection of DFT as shown in Figure 4-1.

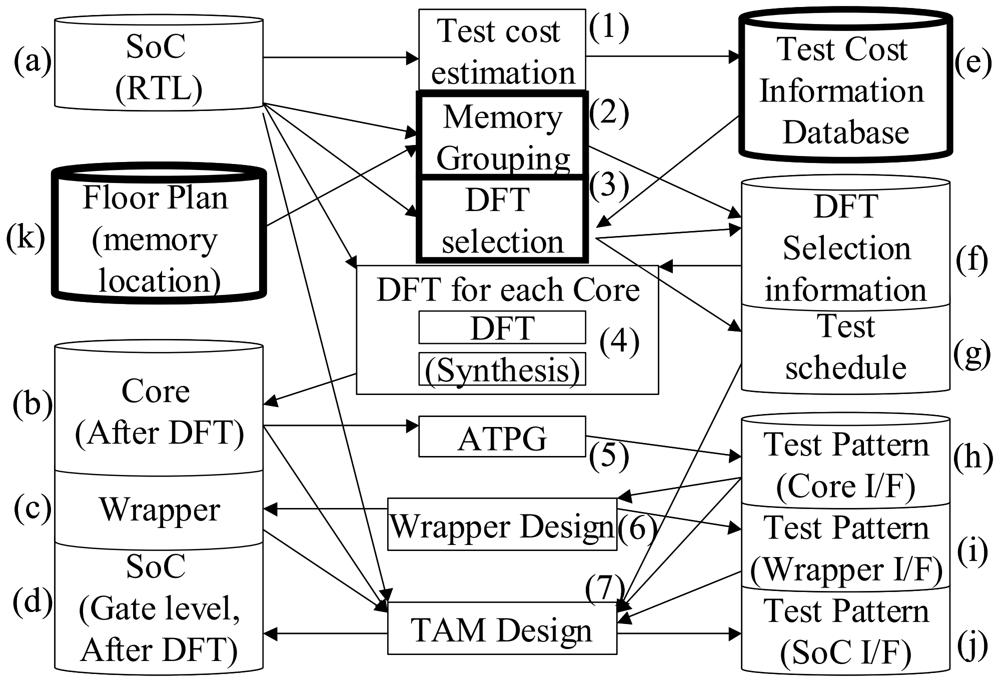

Fig. 4-1 Revised framework of SoC test architecture generation

4-2 Future Work

The development cost of SoCs becomes very high. Therefore, it is very important to estimate the cost as much as possible at an early stage, and to take the trade-off at the cost, the quality, and the development period. So, the test cost estimation step of the framework that presented in this thesis is very important.

Actually, it is not easy to estimate the test cost information with high accuracy at the early stage of the design. As described in chapter 2, test cost information obtained from a past design can be used in a reused portion. However, if a new DFT technique will be proposed in the future, it is necessary to estimate the cost information.

Moreover, if new DFT technique is proposed, we have to check that the problem formulation of this research work should be revised or not. For example, memory repair techniques are not mentioned in this thesis. However, the increasing of the percentage of embedded memory used in SoC as shown in chapter 1 will cause the necessity of memory repair technique. Therefore, the research on a method for reducing memory BIST logic that includes the memory repair technology may be needed.

The increasing of function and the embedded memory of SoC will continue along with the progress of process technology. The test cost will increase rapidly if the effort to decrease it is neglected. Therefore, the methods for reducing the testing cost of SoCs should keep being pursued.

Acknowledgements

I would like to thank my supervisor, Professor Hideo Fujiwara, for his guidance and support during my graduate studies. He gave me a chance to enter this program, and changed my engineer life.

I would also like to thank the Co-supervisor of the thesis committee, Professor Kenichi Matsumoto for his valuable comments.

I am highly thankful to Assistant Professor Tomokazu Yoneda for his many valuable suggestions and comments, continuous guidance. All of the discussions with him are helpful for my research work.

I am highly thankful to Associate Professor Michiko Inoue and Assistant Professor Satoshi Ohtake for their valuable discussions on NS-DFT.

I would like to thank members of Computer Design and Test Lab. (Nara Institute of Science and Technology) for their valuable discussions.

I would like to thank Associate Professor Toshinori Hosokawa of Nihon University. When he had been member of STARC, he had been a research partner of the DFT selection problem. He recommended me to enter the graduate school, and changed my life with Professor Fujiwara.

I am thankful to Professor Tomoo Inoue and Asssociate Professor Hideyuki Ichihara of Hiroshima City University for their valuable discussion and comments.

I am highly thankful to Rafael K. Morizawa, Yasuo Sato (STARC) for their valuable comments about presentation.

I am grateful to Hideyuki Nomoto (Renesas Technology) for his valuable comment about power consumption constraints on products.

I wish to express my gratitude to Hiroshi Date (System JD) and Michiaki Muraoka (SIRIJ) for alluring me in STARC, and having given the chance to get acquainted with a lot of researchers.

Finally, I wish sincerely to express my gratitude to my parents and my elder sister.

References

[1] T. Ono, K. Wakui, H. Hikima, Y. Nakamura and M. Yoshida, "Integrated and automated design-for-testability implementation for cell-based ICs," *Proc. 6th Asian Test Symposium,* pp.122-125, November 1997.

[2] E. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg and C. Wouters, "A structured and scalable mechanism for test access to embedded reusable cores," *Proc. International Test Conference*, pp.284-293, October 1998.

[3] P. Vama and S. Bhatia, "A structured test re-use methodology for core-based system chips," *Proc. International Test Conference*, pp.294-302, October 1998.

[4] N. A. Touba and B. Pouya, "Testing embedded cores using partial isolation rings," *Proc. VLSI Test Symposium,* pp.10-16, May 1997.

[5] L. Whetsel, "An IEEE 1149.1 based test access architecture for ICs with embedded cores," *Proc. International Test Conference*, pp.69-78, November 1997.

[6] M. Nourani and C. A. Papachristou, "Structural fault testing of embedded cores using pipelining," *J. Electronic Testing: Theory and Applications. vol. 15*, pp.129-144, 1999.

[7] I. Ghosh, S. Dey, N. K. Jha, "A fast and low cost testing technique for core-based system-on-Chip," *Proc. 35th Design Automation Conference,* pp.542-547, 1998.

[8] I. Ghosh, S. Dey, and N. K. Jha, "A low overhead design for testability and test generation technique for core-based system-on-a-chip," *IEEE Trans.on CAD, vol.18, No.11*, pp.1661-1676, November 1999.

[9] T. Yoneda, H. Fujiwara, "A DFT Method for core-based systems-on-a-chip based on consecutive testability," *Proc. 10th Asian Test Symposium*, pp.193-198, November 2001.

[10] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip," *J. Electronic Testing: Theory and Applications.vol.18*, pp.213-230, April 2002

[11] Y. Huang, W. T. Cheng, C. C. Tsai, and N. Mukherjee, "Resource allocation and test scheduling for concurrent test of core-based SOC design," *Proc. 10th Asian Test Symposium*, pp.265-270, November 2001.

[12] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Integrated test scheduling, test

parallelization and TAM design," *Proc. 11th Asian Test Symposium*, pp.397-404, November 2002.

[13] H. S. Hsu, J. R. Hung, K. L. Cheng, C. W. Wang, C. T. Huang, and C. W. Wu, "Test scheduling and test access architecture optimization for system-on-chip," *Proc. 11th Asian Test Symposium*, pp.411-416, November 2002.

[14] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "On using rectangle packaging for SOC wrapper/TAM co-optimization," *Proc. VLSI Test Symposium,* pp.253-258, May 2002.

[15] Y. Huang, N. Mukherjee, S. Reddy, C. Tsai, W. Cheng, O. Samman, P. Reuter, and Y. Zaidan, "Optimal core wrapper width selection and SOC test scheduling based on 3-Dimensional bin packing algorithm," *Proc. International Test Conference*, pp.74-82, October 2002.

[16] H. Date, T. Hosokawa, and M. Muraoka, "A SoC test strategy based on a non-scan DFT method," *Proc. 11th Asian Test Symposium*, pp.305-310, November 2002.

[17] S. Nagai, S. Ohotake, and H. Fujiwara, "A DFT method for RTL data paths based on strong testability to reduce test application time," *Technical Report of IEICE DC2002-84*, pp.31-36, February 2003.

[18] B. Koneman, "LFSR-coded test patterns for scan designs," *Proc. European Test Conference*, pp.237-242, March 1993.

[19] I. Bayraktarouglu and A. Orailoglu, "Test volume and application time reduction through scan chain concealment," *Proc. 38th Design Automation Conference,* pp.151-155, June 2001.

[20] A. Benso, S. Di Carlo, G. Di Natale and P. Prinetto, "A programmable BIST architecture for clusters of multiple-port SRAMs," *Proc. International Test Conference*, pp.557-566, October 2000.

[21] B. Nadeau-Dostie, *Design for at-speed test, diagnosis and measurement*, Kluwer Academic Publishers, 2000.

[22] H. Nagamochi and T. Ibaraki, "A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph," *Algorithmica, vol.7*, pp.583-596, 1992.

[23] H. Nagamochi and T. Ibaraki, "Computing the edge-connectivity of multigraphs and

capacitated graphs," *SIAM J. Discrete Mathematics, vol.5*, pp.54-66, 1992.

[24] Charles E. Stroud, *A designer's guide to built-in self-test*, Kluwer Academic Publishers, 2002.

[25] International Technology Roadmap for Semiconductors 2004 Update. (Design) http://public.itrs.net/