PAPER

A Memory Grouping Method for Reducing Memory BIST Logic of System-on-Chips

Masahide MIYAZAKI[†a)], Tomokazu YONEDA[†], *Members*, *and* Hideo FUJIWARA[†], *Fellow*

SUMMARY 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 memory BIST logics were individually added to these various memories, the area overhead would be very high. To reduce the overhead, memory BIST logic must therefore be shared. This paper proposes a memory-grouping method for memory BIST logic sharing. A memory-grouping problem is formulated and an algorithm to solve the problem is proposed. Experimental results show that the proposed method reduced the area of the memory BIST wrapper by up to 40.55%. The results also show that the ability to select from two types of connection methods produced a greater reduction in area than using a single connection method.

key words: *SoC, test scheduling, wrapper, design for test, memory BIST*

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 [1]. 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 [2]. 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 paper, we propose two types of memory-connection methods for BIST wrapper sharing. 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 paper is organized as follows. In Sect. 2, our method for memory BIST logic sharing is described. In Sect. 3, the memory-grouping problem and an algorithm to solve the problem are presented. The experimental results are shown in Sect. 4. Finally, Sect. 5 concludes this paper.

2. Memory BIST Logic Sharing

In this section, we describe our method of BIST logic sharing for single port and word access memory. Figure 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 bypass 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,

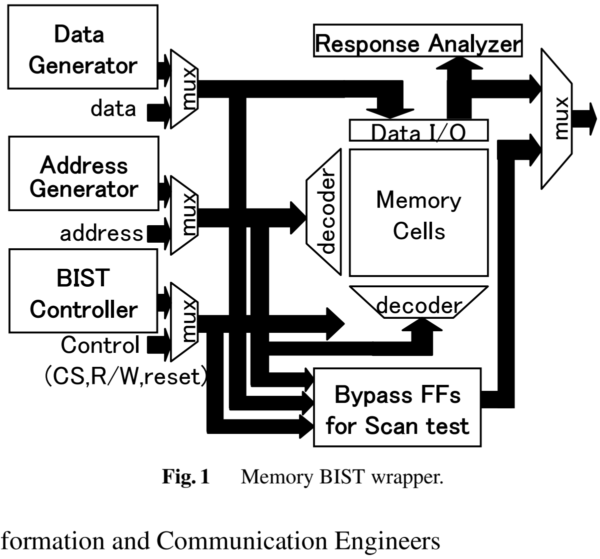


Fig. 1 Memory BIST wrapper.

Manuscript received May 6, 2005.
Manuscript revised August 29, 2005.

[†] The authors are with Nara Institute of Science and Technology, Ikoma-shi, 630–0101 Japan.
a) E-mail: masah-mi@is.naist.jp

DOI: 10.1093/ietisy/e89–d.4.1490

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 paper, we treat the following two memory connection methods for memory BIST logic sharing: parallel connection and serial connection.

Parallel connection can be used to connect memories that have the same number of words. Figure 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 shows an example of four serially connected 8 × 32 word memories. In this example, the four memories are tested as an 8 × 128 word memory. The address generator generates an additional 2 bit 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 8 bit generator and 8 bit response analyzer.

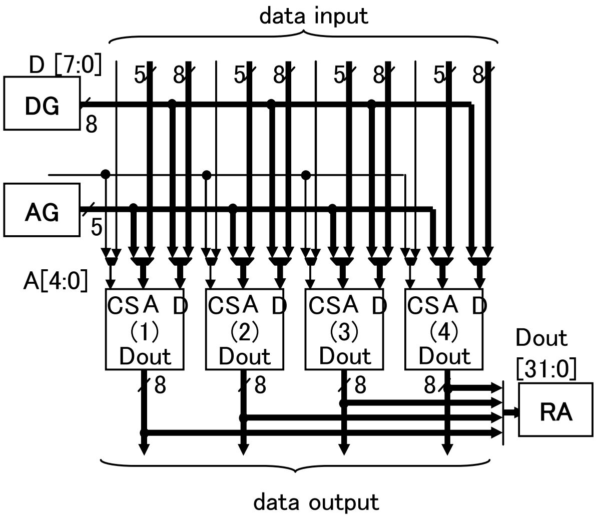


Fig. 2 Parallel connection of memories.

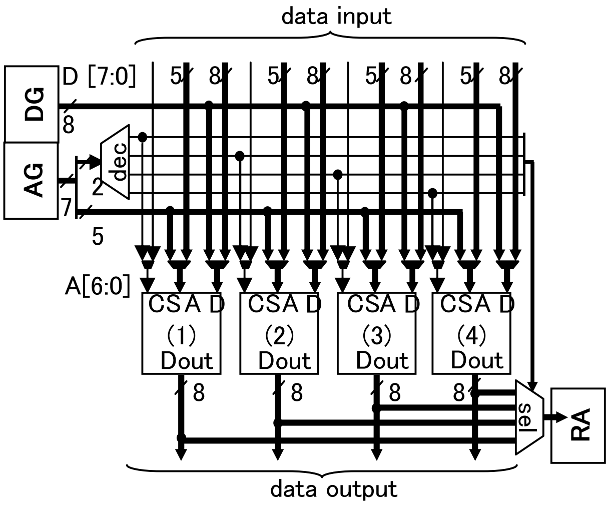


Fig. 3 Serial connection of memories.

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. Memory-Grouping Problem and Algorithm

3.1 Formulation of Memory-Grouping Problem

In this subsection, we present a memory-grouping problem. We assume 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.

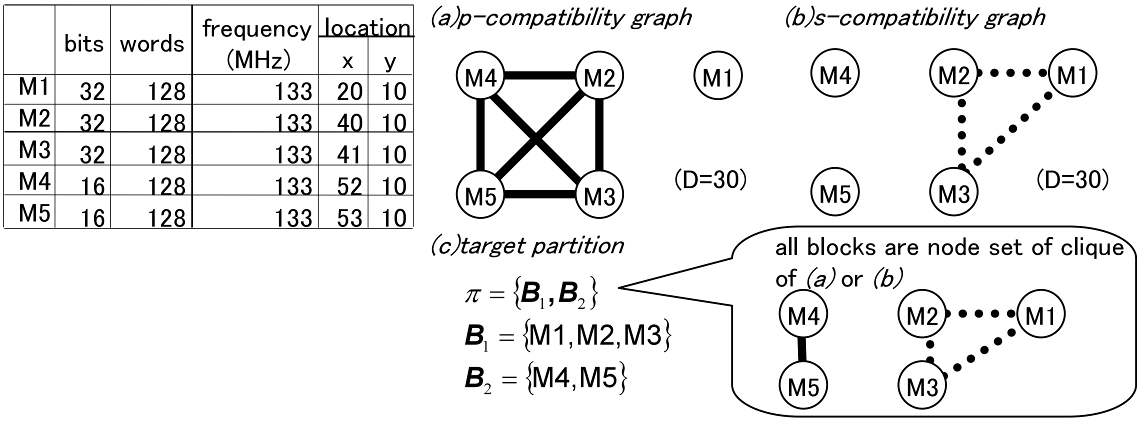
In the same way, a pair of memories $m_i, m_j \in V$ is ***s-compatible*** if they satisfy the following conditions:

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

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

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

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



	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

(a)p-compatibility graph
M4
M2
M5
M3
M1
(D=30)
(b)s-compatibility graph
M4
M2
M1
M5
M3
(D=30)
(c)target partition
$\pi = \{\boldsymbol{B}_1, \boldsymbol{B}_2\}$
$\boldsymbol{B}_1 = \{\mathrm{M1,M2,M3}\}$
$\boldsymbol{B}_2 = \{\mathrm{M4,M5}\}$
all blocks are node set of clique
of (a) or (b)
M4
M5
M2
M1
M3

Fig. 4 Compatibility graphs and our target partition.

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 $\pi = \{B_1, B_2, \ldots B_K\}$ 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 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 Fig. 4 (a) and (b), respectively. Figure 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.

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 [5] 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,

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

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

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

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

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

$$\begin{aligned}\text{Area } S_{Bj} &= 0.75\left(\log_2\left(\sum_{l=1}^{k} w_l\right)\right)^2 + 2k\log_2\left(\sum_{l=1}^{k} w_l\right)\\ &\quad + 25\log_2\left(\sum_{l=1}^{k} w_l\right) + k(\log_2 k)\\ &\quad + 9b_{Bi}k + 14b_{Bi} + 8k + 61\\ b_{Bi} &= b_1 = b_2 = \ldots = b_k \qquad (10)\end{aligned}$$

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

$$\begin{aligned}\text{Test application time } T_{Bj} &= 8 \times \left(\sum_{l=1}^{k} w_l\right)/f_{Bj}\\ &\quad \times (\text{number of background patterns}) \qquad (12)\end{aligned}$$

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 Sect. 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}.

$$S_{total} = \sum_{i=1}^{K} S_{Bi} \qquad (13)$$

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 [6]. 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. 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 Fig. 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: $\boldsymbol{M} = M_i\,(b_i, w_i, p_i, f_i, x_i y_i)$
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:

Maximum distance of memory connection: D
Maximum available peak power of the SoC: P
Maximum test application time of memory: T

Objective:

To minimize S_{total}.
To solve this problem, an algorithm is proposed below.

3.2 Memory-Grouping Algorithm

Figure 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 con-

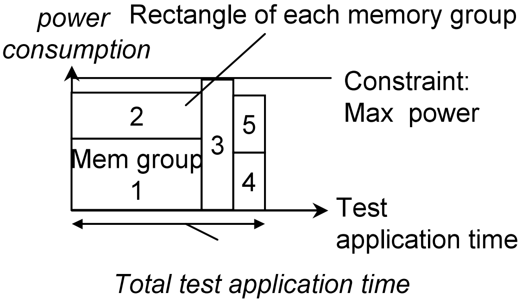


Fig. 5 Test scheduling using rectangle packing.

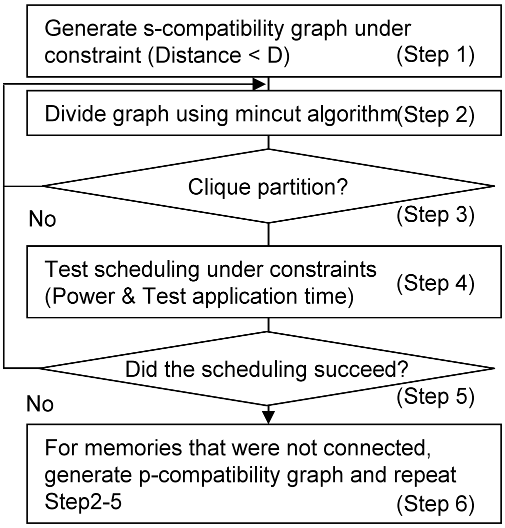

Generate s-compatibility graph under constraint (Distance < D) (Step 1)
Divide graph using mincut algorithm(Step 2)
Clique partition?
No
(Step 3)
Test scheduling under constraints (Power & Test application time) (Step 4)
Did the scheduling succeed?
No
(Step 5)
For memories that were not connected, generate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

Fig. 6 Memory grouping algorithm.

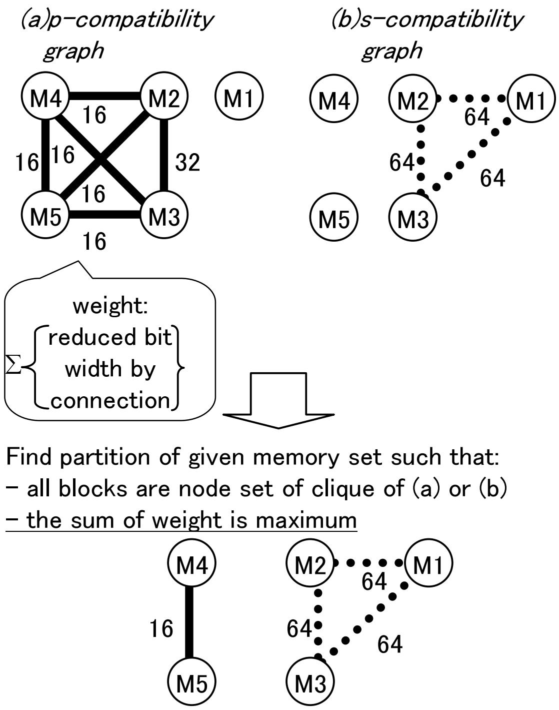

(a)p-compatibility graph
(b)s-compatibility graph
M4
M2
M1
M5
M3
16
16
16
32
16
16
64
64
64
weight:
Σ{reduced bit width by connection}
Find partition of given memory set such that:
– all blocks are node set of clique of (a) or (b)
– the sum of weight is maximum

Fig. 7 Heuristic of graph division.

nection.

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 Fig. 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 Fig. 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 $32 + 32 = 64$. To ensure that the area is reduced as much as possible, the min-cut method [3], [4] is used. 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 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 (line 22–24). Then the algorithm stores p-compatibility graph into G, and collects the blocks that have only one memory into one block (line 25–29). Then, the algorithm searches for the partition that S_{total} is minimized using p-compatibility (line 5–24).

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 $O(V^3 \log V)$.

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 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:

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

The power consumption is proportional to the area.

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. 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 pat-

```
Procedure Memory_Grouping ($\boldsymbol{M}$, $P$, $T$, $D$){
$S_{min}$= the total area of memory BIST wrapper without sharing; maxedgenum=0; edgenum=0;
$\pi$ ={B}, B={$m_1$, $m_2$, ...$m_n$};   $\pi_{tmp} = \phi$ ;   $\pi_{next} = \phi$;   $\pi_{best} = \phi$; $\pi_{s-compatible} = \phi$;
$G_s$ = s-compatibility_graph of B; $G_p$ = p-compatibility_graph of B;
$G = G_s$;
loop:
Construct a set of graph $C_{all}$={ $G_i$| $G_i$ is induced graph on G by $B_i \in \pi$ }
for( {$B_i \in$ ( $\pi - \pi_{s-compatible}$ )|which includes two or more memories}){
  delete min-cut edge from $G_i$, make a set of graph $C_{min}$={$G_{i1}$,$G_{i2}$ };
  $C_{all}$= ($C_{all}$ - $G_i$) $\cup$ $C_{min}$;
  Set edgenum= $\sum_j$ (the number of edges of $G_j \in C_{all}$);
  Set a partition $\pi_{tmp}$ ={$B_j$| vertex set of $G_j \in C_{all}$ };  if all $G_j$ are clique,calculate $S_{total}$ of $\pi_{tmp}$
  if(( $\forall G_j \in C_{all}$ , $G_j$ is clique )$\wedge$ ($S_{min}$>$S_{total}$ of $\pi_{tmp}$ )){
    calculate $T_{total}$=Schedule(P, {$P_{Bj}$}, {$T_{Bj}$});
    if((Schedule succeeded) $\wedge$($T_{total} \leq$ T)){
      $S_{min} = S_{total}$;  $\pi_{best} = \pi_{tmp}$ ;
    }
  }
  if(edgenum > maxedgenum$\wedge$((Schedule failed, or $T_{total} \leq$ T)$\vee$
     ( $\exists G_j \in C_{all}$ , $G_j$ is not a clique))){
      $\pi_{next} = \pi_{tmp}$ ; maxedgenum=edgenum;
  }
}
if( $\pi_{next} \neq \phi$ ){
  $\pi = \pi_{next}$ ; $\pi_{next} = \phi$ ; go to loop;
}
else if(G=$G_s$){
  G = $G_p$;
  $\pi_{s-compatible}$ ={$B_j \in \pi_{best}$ | which includes two or more memories };
  $B_s$= $\bigcup_k B_k$   ($B_k \in$ ( $\pi_{best} - \pi_{s-compatible}$ ));
  $\pi$ ={$B_s$}$\cup \pi_{s-compatible}$ ; go to loop;
}else{end;}
```

Fig. 8 Memory grouping algorithm. (pseudo code)

Table 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

terns = 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: $T = 300$

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); and (5) exhaustive search (memory BIST logic was shared using only parallel connection after an exhaustive search). Table 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 shows the total area of memory BIST wrappers using

Table 2 Area of memory-BIST logic.

#of	not	Proposed algorithm			
mem	shared	P only	S only	S&P	exhaustive
3	2289	1967	1660	1660	1660
4	2913	2591	2284	2284	2284
5	3537	2893	2279	2279	2279
6	4203	3559	3044	2722	2415
7	4869	3863	3690	3368	2540
8	5535	4529	3719	3397	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	

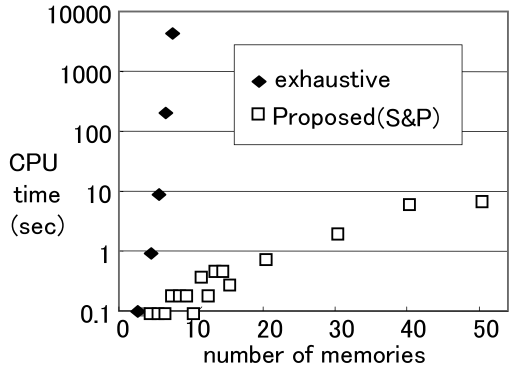


Fig. 9 CPU time for memory grouping program.

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.

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 show that the memory BIST logic sharing technique reduced the area of the BIST wrappers by between 21.59 and 47.83% as minimum solutions. However, the technique achieved only 64.45% of the minimum solution in these cases, so there is room for improving the quality of the solution.

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

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.

As future work we will investigate improving the quality of the solution and minimizing the test application time.

Acknowledgments

Authors would like to thank Prof. Michiko Inoue and Prof. Satoshi Ohtake and members of Computer Design and Test Lab. (Nara Institute of Science and Technology) for their valuable discussions.

References

[1] 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, Oct. 2000.

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

[3] 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.

[4] H. Nagamochi and T. Ibaraki, “Computing the edge-connectivity of multigraphs and capacitated graphs,” SIAM J. Discrete Mathematics, vol.5, pp.54–66, 1992.

[5] C.E. Stroud, A designer’s guide to built-in self-test, Kluwer Academic Publishers, 2002.

[6] 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.

[7] 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, Oct. 2002.

Masahide Miyazaki received the B.E. and M.S. degrees in Space Science from Nagoya University, Nagoya, Japan, in 1990 and 1992, respectively. In 1992, he joined Hitachi Ltd. He has worked on sequential redundancy, and design for testability. In 2002, he went to Semiconductor Technology Academic Research Center (STARC) as a Hitachi assignee, where is currently working on testing for system on a chip. He entered the graduate program of the Graduate School of Information Science, Nara Institute of Science and Technology in the Autumn of 2003.

Tomokazu Yoneda received the B.E. degree in information systems engineering from Osaka University, Osaka, Japan, in 1998, and M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan, in 2001 and 2002, respectively. Presently he is an Assistant Professor in Graduate School of Information Science, Nara Institute of Science and Technology. His research interests are VLSI CAD, design for testability and SoC testing. He is a member of the IEEE Computer Society.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology in 1993. In 1981 he was a Visiting Research Assistant Professor at the University of Waterloo, and in 1984 he was a Visiting Associate Professor at McGill University, Canada. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He received the IECE Young Engineer Award in 1977, IEEE Computer Society Certificate of Appreciation Award in 1991, 2000 and 2001, Okawa Prize for Publication in 1994, IEEE Computer Society Meritorious Service Award in 1996, and IEEE Computer Society Outstanding Contribution Award in 2001. He is an advisory member of IEICE Trans. on Information and Systems and an editor of IEEE Trans. on Computers, J. Electronic Testing, J. Circuits, Systems and Computers, J. VLSI Design and others. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society and a fellow of the IPSJ (the Information Processing Society of Japan).