IEEE the 11th Asian Test Symposium (ATS'02), pp. 128-133, Nov. 2002.

A Scheduling Method in High-Level Synthesis for Acyclic Partial Scan Design

Tomoo Inoue[†], Tomokazu Miura[*†], Akio Tamura[†], and Hideo Fujiwara[‡]

[†] Faculty of Information Sciences, Hiroshima City University
3-4-1 Ozukahigashi, Asaminami, Hiroshima 731-3194, Japan, Ph./Fax:+81-82-830-1575

[‡] Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0101, Japan, Ph.:+81-743-72-5220, Fax:+81-743-72-5229

{*tomoo,tamura*}*@im.hiroshima-cu.ac.jp*,*fujiwara@is.aist-nara.ac.jp*

Abstract. *Acyclic partial scan design is an efficient DFT method. This paper presents a scheduling method for reducing the number of scan registers for acyclic structure. In order to estimate the number of scan registers during scheduling, we propose provisional binding of operational units, and show a force-directed scheduling algorithm with the provisional binding. Experimental results show that the number of scan registers in the resulting RTL datapaths can be reduced by our method combined with the binding algorithm for acyclic partial scan.*

1 Introduction

Design-for-testability (DFT) at register-transfer level (RTL) is an important approach to the reduction of testing cost. RTL DFT is suitable for today's VLSI design style — The design modification for testability is transparent for RTL designers, and the area/delay penalty by the DFT is alleviated by logic synthesis.

Acyclic partial scan design is an efficient DFT method. In this DFT, some of flip-flops in a sequential circuit are replaced by scan flip-flops so that the resultant kernel circuit becomes acyclic. Test generation for an acyclic sequential circuit can be performed by applying a *combinational* test generation algorithm to its *time-expansion model* [1], and hence a complete test set for an acyclic sequential circuit can be obtained efficiently. Such structure-based DFTs [1]-[4] can be applied to RTL designs since structural properties in RTL designs are generally succeeded to logic level circuits by logic synthesis. An efficient algorithm for finding a minimum set of scan registers that break all feedback loops in a circuit has been proposed in [5]. Thus, we can obtain testable circuits with low hardware overhead efficiently.

High-level synthesis refers to transforming an abstract behavioral description into an RTL circuit. Taking DFTs applied to the resultant RTL circuits into account during high-level synthesis, i.e., *high-level synthesis for testability* can reduce the area/delay penalty, and bring out the effect of the RTL DFTs [6]. Many techniques on high-level synthesis for partial scan design have been proposed [7]-[12].

In this paper, we consider a method of high-level synthesis for acyclic partial scan design. High-level synthesis mainly consists of two tasks: *scheduling* and *binding*. Takasaki *et al.* [12] proposed a binding algorithm for minimizing the number of scan registers so as to make the kernel acyclic for a given *scheduled* data flow graph (DFG) or behavior description while keeping the area/performance optimality. Experimental results in [12] show that the binding method can minimize the number of scan registers for many scheduled DFGs. In this paper, we propose a scheduling method for reducing the number of scan registers for acyclic structure. In order to estimate the number of scan registers during scheduling, we propose *provisional binding* of operational units, and show a force-directed scheduling algorithm [13] with the provisional binding. Experimental results show that the combination of the scheduling algorithm with the provisional binding and the binding algorithm [12] can further reduce the number of scan registers for acyclic structure compared to that of the scheduling algorithm without the provisional binding and the binding algorithm for some example DFGs.

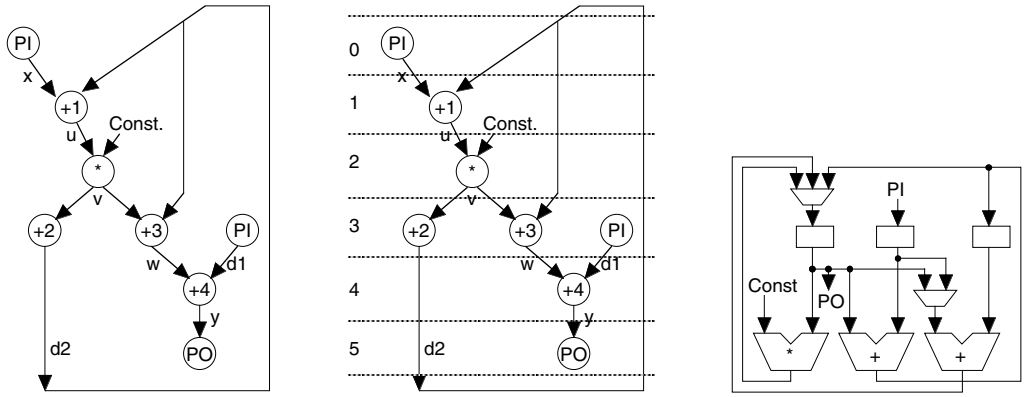
2 High-Level Synthesis for Acyclic Structure

2.1 High-level synthesis flow

High-level synthesis (HLS) transforms a behavioral information, which is represented by a *data flow graph* (DFG) (Fig. 1(a)), into a register-transfer level (RTL) design (Fig. 1(c)). In a DFG, a vertex represents an operation with a type (adder, multiplier, and so forth), and an edge represents a variable. Our HLS system derives an optimal RTL datapath in which the number of resources (operational units and

[*] He is currently with Matsushita Electric Industrial Co., Ltd., Japan.

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 IEEE



(a) Data flow graph: G_D (b) Scheduled DFG: G_{sD} (c) RTL datapath for G_D

Figure 1. Input and output of high level synthesis.

registers) are minimum for a given DFG with a *latency* constraint, which is an upper bound of the execution time. In addition, it aims to minimize the number of scan registers required for cycle breaking in the resultant RTL circuits.

HLS mainly consists of two tasks: *scheduling* and *binding*. The former, scheduling, is to minimize the upper bound of the number of operational units for each operation type (adder, multiplier, and so on) under a latency constraint by assigning each operation to an appropriate control step. The result of the scheduling is represented by a *scheduled* DFG (Fig. 1(b)).

The latter binding procedure is further divided into operational unit binding and register binding. Here we consider that operational unit binding is followed by register binding. In these procedures, each operation (variable) in the DFG *scheduled* by the above procedure is assigned to an operational unit (register) in the resultant RTL datapath, i.e., the binding determines which each operation (variable) *shares* an operational unit (register) with. As a result, an optimal datapath with minimum numbers of operational units and registers is obtained (Fig. 1(c)).

2.2 Binding algorithm

The optimal operational unit binding problem is reduced to the minimum clique partitioning problem for a *compatibility graph* of operations. In a compatibility graph G_C (Fig. 2) for a scheduled DFG G_{sD}, a vertex corresponds to an operation, and an edge (v_1, v_2) denotes that operations v_1 and v_2 are *compatible*, i.e., they can share an identical operation. Note that two operations v_1, v_2 are compatible when they are assigned to different control steps in the scheduled DFG G_{sD}. A clique in compatibility graph G_C means that all the operations in the clique can share one identical operations, and hence the goal of binding is to partition the compatibility graph with a minimum number of cliques. An efficient heuristic algorithm for the minimum clique partitioning problem has been proposed [14].

Recall that our goal is to minimize the number of scan registers for acyclic partial scan while minimizing the number of resources under a given latency constraint. In order to take the number of scan registers into account during the above-mentioned minimum clique partitioning, by focusing on the fact that sharing may cause a loop, Takasaki *et al.* [12] represented the strength or possibility of the requirement for scan registers as a *weight* on an edge in the compatibility graph, and presented an algorithm for finding a minimum weighted clique partitioning.

In [12], edges (v_1, v_2) in compatibility graph G_C are classified into three cases based on the relation of two operations v_1 and v_2 in the scheduled DFG G_{sD}, and accordingly edges are weighted as follows[1].

v_1 and v_2 are adjacent. The sharing must make a self-loop. A self-loop must need a scan register, and therefore the weight is *large*.

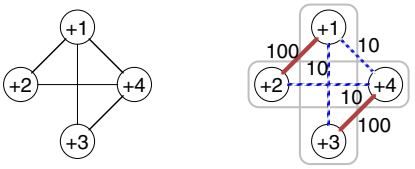
v_1 and v_2 are not adjacent, but reachable. This makes a (global) feedback loop. There must exist at least one scan register on a path between v_1 and v_2. This, however, implies that an assignment of *some* variable on the path to a scan register is sufficient. Hence, the weight is relatively *small*.

There is no path between v_1 and v_2. In this case, no loop is made, and hence the weight is *zero*.

Based on the above weighting, the weight of a clique is defined as the sum of weights of edges in the clique. A heavy clique is regarded as a sharing that requires scan registers strongly. Thus, the problem (minimizing the number of scan registers with the minimum number of resources) is solved by finding a minimum clique partitioning such that the sum of weights of the cliques is minimum.

[1] In [12], the weight is specified in more detail according to the scheduled control steps of operations and the number of variables between them. However, we need not such a specified weight in the following discussion,

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 IEEE



(a) Compatibility graph G_C.

(b) Minimum weighted partitioning.

Figure 2. Compatibility graph: G_C for adders in G_D.

Example 1: Fig. 2(a) shows the compatibility graph G_C for adders in the scheduled DFG G_{sD}. Fig. 2(b) shows the weights of edges and the optimal clique partitioning. In this example, large and small weights are given by 100 and 10, respectively, for the simplicity. The number of cliques is two, and hence, the number of adders in the resultant RTL circuits becomes two. The sum of weights of cliques is $10+10=20$, which is the minimum. Note that the weight of any other clique partitioning whose number of cliques is also two is larger than 20 (e.g., $\{\{+1,+2\},\{+3,+4\}\}$ derives weight 200). □

Similarly, register binding is performed, and thus an optimal RTL datapath can be obtained.

3 Scheduling Algorithm with Provisional Binding

Recall that the goal of our HLS is to reduce the number of scan registers without obstructing the resource minimization. Hence, when the scheduling algorithm finds more than one candidates of operation assignment for the resource minimization, it must select the best for scan register reduction of all the candidates. It is, however, difficult to know the accurate number of scan registers for acyclic structure during scheduling since the specific structure of resultant RTL circuits is determined after the completion of the binding procedure, which succeeds the scheduling. Therefore, as a method for estimation of the number of scan registers during the schedule algorithm, we propose *provisional operational binding*.

In the following discussion, as an example of the applications of provisional binding, we present a *force directed scheduling* algorithm [13] with our provisional binding. This provisional binding can be applied to other scheduling algorithms.

3.1 Force-directed scheduling

In the scheduling procedure, some operations are assigned to control steps uniquely due to a latency constraint. For example, when a latency constraint for DFG G_{D1} (Fig. 3) is 5, operations $+1,+4,+6$ and $*$ are uniquely assigned to steps 1,2,3 and 4, respectively. On the other hand, the remaining operations (highlighted in Fig. 3) still have flexibilities on control steps to be scheduled. For example, operation $+2$ can be assigned to either step 1 or step 2. The difference $t_e-t_s(t_e=2,t_s=1)$ of the start and end of the time interval $[t_s,t_e]$ where an operation v can be scheduled is said to be the *mobility* of v. In this case, the mobility of $+2$ in Fig. 3 is 1. Note that the mobility of a scheduled operation is zero. Thus, the scheduling procedure is to determine appropriate control steps for all the operations whose mobilities are not zero.

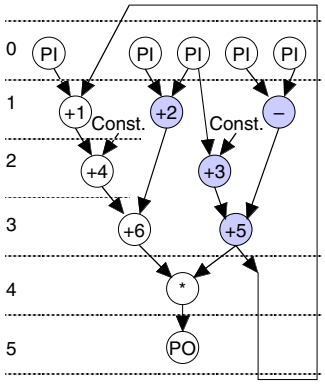


Figure 3. Data flow graph: G_{D1}.

The *force-directed scheduling* (FDS) algorithm [13] is known to be an efficient algorithm for finding an optimal scheduling. In this algorithm, it is supposed that operations are connected to one another by a *spring*, and some *force* is exerted on them. The force changes according to the assignment of operation to control steps[2], and a balanced status (i.e., a case where the whole force is minimum) is regarded as an optimal scheduling.

The sketch of the FDS algorithm is as follows.

- For all operations v whose mobilities are not zero, force $f(v,t)$ is calculated for all steps $t\in[t_s,t_e]$ where v can be scheduled. The method for precise calculation of forces is omitted here (See [13]). If a control step t is crowded by the operations whose type is the same as v (i.e., the probability that many operations are assigned to t is high), the force $f(v,t)$ becomes large, and accordingly operation v will push out of step t. Otherwise, i.e., when step t is not crowed, $f(v,t)$ becomes small, and consequently operation v will be pull into step t.
- Select a minimum force $f(v,t)$, and operation v is assigned to step t.

and hence the precise definition is omitted here.

[2] Although the force of the original FDS algorithm can take various factors in optimization (e.g., the number of registers) into account, here we make the force represent only the factor concerned with the number of operational units, for the sake of simplicity.

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 IEEE

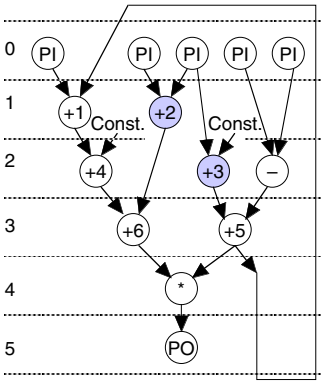


Figure 4. Partially-scheduled DFG: G'_{sD1}.

- According to the assignment, update the mobility of each operation that is not assigned to any control step, and repeat the above procedures until all operations are scheduled.

Example 2: Consider DFG G_{D1} in Fig. 3. At the beginning of the FDS, operations $+2, +3, +5$ and $-$ all have mobility 1, and they have not been scheduled. Note that the control steps of the other operations are uniquely determined. Forces $f(-,2)$ and $f(+5,3)$ are the minimum for all time steps of all the unscheduled operations, and the FDS algorithm chooses either of them. For example, if the former $(-,2)$ is selected, operation $-$ is assigned to step 2. □

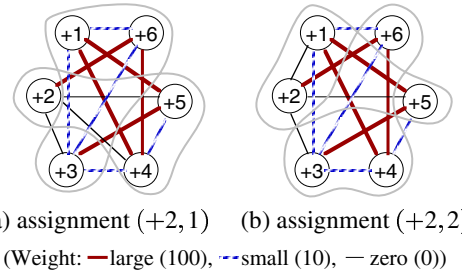
3.2 FDS with provisional binding

As mentioned above, each operation is assigned to an appropriate control step at every iteration in the FDS algorithm. Here, suppose that a *partially-scheduled* DFG G'_{sD1} shown in Fig.4 is obtained in the progress of FDS. In G'_{sD1}, operations $+2$ and $+3$, which are highlighted, are not scheduled, and they both have mobility 1. The forces $f(+2,1)$, $f(+2,2)$, $f(+3,1)$ and $f(+3,2)$ are all the same and minimum. In such a case, i.e., when there exist two or more candidates for the best assignment with forces, we apply the *provisional binding* to the candidates to break the tie.

Let A be a set of assignments (v,t) whose force $f(v,t)$ are minimum in a partially-scheduled DFG G'_{sD}. For each assignment $(v,t) \in A$, the weight $w_p(v,t)$ is calculated by the followings.

- Make a provisional compatibility graph $G_C(G'_{sD},v,t)$ for another partially-scheduled DFG $G'_{sD}(v,t)$ obtained by applying assignment (v,t) to G'_{sD}. In $G_C(G'_{sD},v,t)$, a vertex corresponds to an operation, and an edge (v_1,v_2) denotes that operations v_1 and v_2 are compatible *provisionally*. Here, two operations v_1 and v_2 are regarded as (provisionally) compatible not only when they are assigned to different control steps, but also when either of two operations has non-zero mobility (or is unscheduled). This is because an operation v_1 whose mobility is not zero can share an operational unit with the other v_2 by assigning v_1 to step t_1 which is different from step t_2 where v_2 is assigned.
- Each edge in $G_C(G'_{sD},v,t)$ is weighted in the same way as the binding method of [12] provided that the specific weight is omitted as explained in the previous section since the specification of weights is not possible before the completion of scheduling, and such a rough weighting is considered to be sufficient for estimating scan requirement.
- According to the above-mentioned weighting, the minimum weighted clique partitioning problem is solved by the method [12], and let weight $w_p(v,t)$ for the assignment (v,t) in partially-scheduled DFG G'_{sD} be the weight of the clique partitioning (the sum of clique weights).

We choose an assignment (v,t) whose weight $w_p(v,t)$ is a minimum based on the above calculation as a solution for breaking the tie of the minimum force selection.



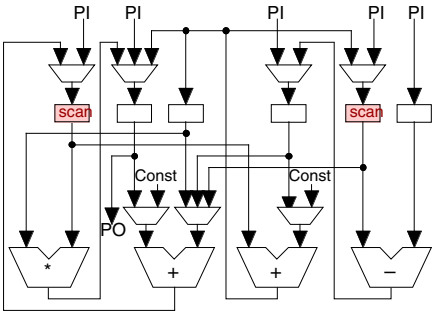
(a) assignment $(+2,1)$ (b) assignment $(+2,2)$

(Weight: — large (100), - - small (10), ··· zero (0))

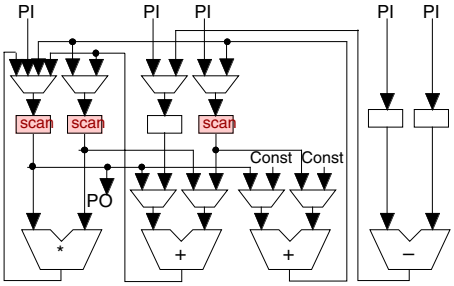
Figure 5. Provisional compatibility graphs for G'_{sD1}.

Example 3: Figs. 5 (a) and (b) show the provisional compatibility graph $G_C(G'_{sD1},+2,1)$ and $G_C(G'_{sD1},+2,2)$ for assignments $(+2,1)$ and $(+2,2)$ in G'_{sD1}, respectively. The wights $w_p(+2,1)$ and $w_p(+2,2)$ are 40 and 220, respectively. Similarly, the others $w_p(+3,1)$ and $w_p(+3,2)$ are 230 and 40, respectively. Assignments $(+2,1)$ and $(+3,2)$ are minimum, and hence assignment $(+2,1)$ is selected here. Then, operation $+2$ is scheduled at step 1 by force-directed scheduling. Note that assignment $(+3,2)$ results in the same scheduling as $(+2,1)$. After updating mobilities and forces, the remaining unscheduled operation is $+3$ only, and accordingly $+3$ is assigned to step 2 with minimum force. As a result, a (completely) scheduled DFG G_{sD1} is obtained. By applying the binding algorithm [12] to G_{sD1}, we can obtain an RTL datapath shown in Fig. 6(a), which requires two scan registers for acyclic structure. Fig. 6(b) shows an RTL datapath obtained by assignment $(+2,2)$ instead of $(+2,1)$ mentioned above and applying the same

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 IEEE



(a) RTL corresponding to assignment $(+2,1)$



(b) RTL corresponding to assignment $(+2,2)$

Figure 6. RTL datapaths.

Table 1. DFG characteristics.

DFG	#PIs	#POs	#ops	#vars
G_{D1}	5	1	8	14
G_{D2}	5	3	11	25
Lwf	2	1	5	7
Paulin	4	3	10	11
Jwf	1	1	17	20
Iir	1	1	17	22
Ewf	1	1	34	38

binding algorithm. The RTL datapath corresponding to $(+2,2)$ requires three scan registers for acyclic structure. Thus, we can reduce the number of scan registers for acyclic structure by scheduling with provisional binding. □

4 Experimental Results

A force-directed scheduling algorithm in which our provisional binding is embedded was implemented, and applied to several benchmark DFGs. For the experiments, a workstation SUN Ultra 10 (UltraSPARC-IIi 440MHz, 1GB Memory) was used. Table 1 shows the characteristics of benchmark DFGs. Columns #PIs, #POs, #ops and #vars denote the numbers of operations, primary inputs, primary outputs and variables, respectively. The first DFG G_{D1} corresponds to the example used in the pervious section (Fig. 3).

In order to analyze the effect of our method, we used not only (SS) our proposed scheduling algorithm (force-directed scheduling algorithm with the provisional binding, i.e., minimum clique partitioning with *minimum* weight), but also (S) the force-directed scheduling algorithm with minimum clique partitioning whose weight is *maximum*. The latter scheduling (S) is considered to aim to raise the number of scan registers obtained by the succeeding binding algorithm. In the same way as the previous section, in the schedulings (SS) and (S), only the basic core of underlying force-directed scheduling algorithm is used, i.e., the force represents the factor concerned with only the number of operational units. Further, we used two binding algorithms: (SB) the binding algorithm [12] for acyclic partial scan (mentioned in Sect. 2.2, i.e., minimum clique partitioning with *minimum* weight) and (B) the binding algorithm by minimum clique partitioning with *maximum* weight. Thus, we applied all the combinations (SS+SB), (SS+B), (S+SB) and (S+B) to the benchmark DFGs, and calculated the minimum number of scan registers that are required for acyclic structure (i.e., for breaking all feedback loops) in the resultant RTL circuits.

Table 2 shows the resulting RTL datapaths obtained by the combinations of scheduling and binding algorithms. The number Latency under each DFG name denotes a given latency constraint, which is minimum for the DFG. Columns #ops, #regs, #scans and #muxs denote the numbers of operational units and registers, the minimum number of scan registers for acyclic structure and the number of two-input multiplexors, respectively. The numbers of operational units and registers are minimum under the latency constraint for all DFGs. The computation time required for scheduling is small: the time required by SS for Ewf was about 0.3 second, which is the largest of all the schedulings.

From this table we can see that, by comparing SS+SB (our method) to S+SB, our scheduling method with the provisional binding reduces the number of scan registers. Furthermore, the number of scan registers by SS+SB is the smallest of all the algorithms. On the other hand, the number of scan registers by SS+B is not smaller than that by S+SB for any DFG, i.e., our scheduling method without the binding algorithm [12] for acyclic partial scan is not effective. This is because our scheduling with provisional binding performs on the premise that the following binding algorithm aims to the reduction in scan registers.

Note that the number of multiplexors in a resultant RTL circuit is important for area/ performance as well as the number of scan registers. Hence, we pursued the number of multiplexors as shown in the rightmost column of Table 2. From these results, we can see that our synthesis method SS+SB can also reduce the number of multiplexors while reducing the number of scan registers for acyclic structure, even though neither SS nor SB takes the number of multiplexors into account. Recall that both of our scheduling SS

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 IEEE

Table 2. Results: RTL datapaths.

DFG (Latency)	Method	#ops	#regs	#scans	#muxs
G_{D1} (5)	**SS+SB**	4	6	**2**	8
	SS+B			4	8
	S+SB			3	10
	S+B			6	11
G_{D2} (5)	**SS+SB**	4	6	**2**	11
	SS+B			6	14
	S+SB			3	12
	S+B			6	16
Lwf (5)	**SS+SB**	3	4	**1**	4
	SS+B			4	4
	S+SB			2	4
	S+B			4	6
Paulin (5)	**SS+SB**	4	6	**4**	8
	SS+B			6	10
	S+SB			4	8
	S+B			6	10
Jwf (9)	**SS+SB**	4	7	**4**	16
	SS+B			7	16
	S+SB			5	16
	S+B			7	16
Iir (8)	**SS+SB**	4	7	**4**	13
	SS+B			7	16
	S+SB			4	13
	S+B			6	16
Ewf (16)	**SS+SB**	4	10	**6**	31
	SS+B			10	32
	S+SB			8	30
	S+B			10	34

and binding SB aim to reduce feedback loops by sharing operational units and registers appropriately. Accordingly we can consider that many paths that make loops share one path, and consequently the number of paths coming into a resource (operational unit or register) decreases successfully. It is a future work to analyze this phenomenon in more detail.

Thus, the combination of SS+SB results in the most effective in reducing scan registers for acyclic structure.

5 Conclusions

In this paper, we presented a scheduling algorithm for reducing the number of scan registers for acyclic structure. In order to estimate the number of scan registers during scheduling, we proposed *provisional binding* of operational units. Experimental results show that a force-directed scheduling algorithm in which our provisional binding is embedded can reduce the number of acyclic scan registers while keeping the area/performance optimality.

Acknowledgments Authors would like to thank Profs. Yoshinori Matsuura and Hideyuki Ichihara of Hiroshima City University for their helpful disscussions. This work was supported in part by Hiroshima City University under the HCU Grant for Special Academic Research (General Studies) and by Foundation of Nara Institute of Science and Technology under the Grant for Activity of Education and Research.

References

[1] T. Inoue, D. K. Das, C. Sano, T. Mihara, and H. Fujiwara, "Test generation for acyclic sequential circuits with hold registers," *IEEE/ACM Proc. Int'l Conf. on Computer Aided Design*, 2000, pp.550-556.

[2] R. Gupta, R. Gupta and M. A. Breuer, "The BALLAST methodology for structured partial scan design," *IEEE Trans. on Comput.*, Vol.39, No.4, pp.538-544, April 1990.

[3] R. Gupta and M. A. Breuer, "Partial scan design of register-transfer level circuits," *Journal of Electronic Testing: Theory and Applications*, Vol.7, pp.25-46, 1995.

[4] T. Inoue, T. Hosokawa, T. Mihara, and H. Fujiwara, "An optimal time expansion model based on combinational test generation for RT level circuits," *Proc. IEEE Asian Test Symp.*, 1998, pp.190-197.

[5] S. T. Chakradhar, A. Balakrishman, and V. D. Agrawal, "An exact algorithm for selecting partial scan design," *Proc. Design Automation Conf.*, 1994, pp.81-86.

[6] M. Inoue and H. Fujiwara, "An approach to test synthesis from higher level," Integration, the VLSI journal 26, pp.101-116, 1998.

[7] T. C. Lee, N. K. Jha, and W. H. Wolf, "Behavioral synthesis of highly testable data paths under the non-scan and partial scan environments," *Proc. Design Automation Conf.*, pp.292-297, 1993.

[8] M. Potkonjak, S. Dey, and R. K. Roy, "Behavioral synthesis of area-efficient testable designs using interaction between hardware sharing and partial scan," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 9, pp.1141-1154, 1995.

[9] A. Mujumdar, R. Jain, and K. Saluja, "Behavioral synthesis of testable designs," *Proc. IEEE Int'l. Symp. on Fault-Tolerant Computing*, pp. 436-445, 1994.

[10] V. Fernandez and P. Sanchez, "Partial scan high-level synthesis," *Proc. European Design and Test Conf.*, pp.481-485, 1996.

[11] M. L. Flottes, R. Pires, B. Rouzeyre and L. Volpe, "Low cost partial scan design: a high level synthesis approach," *IEEE Proc. VLSI Test Symp.*, pp.332-340, 1998.

[12] T. Takasaki, T. Inoue, and H. Fujiwara, "A high-level synthesis approach to partial scan design based on acyclic structure," *IEEE Proc. Asian Test Symp.*, 1999, pp.309-314.

[13] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASIC's," *IEEE Trans. on Computer-Aided Design*, Vol.8, No.6, pp.661-679, June 1989.

[14] G. De Micheli, *Synthesis and optimization of digital circuits*, McGraw-Hill, 1994.

Proceedings of the 11 th Asian Test Symposium(ATS'02)
1081-7735/02 $17.00 © 2002 **IEEE**