IEICE TRANS. FUNDAMENTALS, VOL.E91–A, NO.4 APRIL 2008

PAPER Special Section on Selected Papers from the 20th Workshop on Circuits and Systems in Karuizawa

Novel Register Sharing in Datapath for Structural Robustness against Delay Variation

Keisuke INOUE[†a)], ***Student Member***, **Mineo KANEKO**[†], ***and*** **Tsuyoshi IWAGAKI**[†], ***Members***

SUMMARY As the feature size of VLSI becomes smaller, delay variations become a serious problem in VLSI. In this paper, we propose a novel class of robustness for a datapath against delay variations, which is named structural robustness against delay variation (SRV), and propose sufficient conditions for a datapath to have SRV. A resultant circuit designed under these conditions has a larger timing margin to delay variations than previous designs without sacrificing effective computation time. In addition, under any degree of delay variations, we can always find an available clock frequency for a datapath having SRV property to operate correctly, which could be a preferable characteristic in IP-based design.

key words: *datapath synthesis, delay variation, register assignment, setup and hold constraints*

1. Introduction

With the advance of process technologies, the feature size of VLSI becomes smaller, and switching delay decreases. As the operation speed becomes higher, on the other hand, delay variations caused by the fluctuation of process parameters, the change of the temperature, supply voltage noise, coupling noise, etc., have become a serious problem [1]–[3].

There are several reports on this problem from different points of view. In order to decrease the timing margin, methods for estimating delay variations more precisely have been studied [4]–[6]. In [7] the authors have addressed the problem to correct timing violations after manufacturing by using programmable delay elements (PDEs). In many cases, the hold constraint, which requests that an operation result has to be kept as it is until it is certainly latched by a register, becomes critical for a correct latch of a signal under delay variations, and specially configured registers have been proposed to try to ensure the hold constraint [8]. Compensation of the minimum path delay is also a candidate technique to ensure the hold constraint.

In this paper, we propose a novel class of robustness against delay variations for a datapath. As the first attempt to study the robustness against delay variations, we focus on the functional aspect of a circuit, which is mainly determined by its structure and its control schedule. In particular, we characterize a datapath which operates correctly under delay variations as a datapath which always has, under any given finite delay variations, a feasible clock frequency with which the datapath operates correctly, and we propose design conditions for a datapath to have such property. It is shown that, compared with conventional designs, a resultant datapath designed with our property has a large timing margin to delay variations without sacrificing effective computation time.

The rest of this paper is organized as follows. Section 2 describes the concept of a novel class of robustness against delay variations for a datapath, which is named structural robustness against delay variation (SRV), and also describes sufficient conditions for a datapath to have SRV. Section 3 presents a feasible register assignment based on SRV. In Sect. 4, we compare our method with some existing techniques. Application examples are presented in Sect. 5. Section 6 treats a minimum register assignment for SRV, and shows several computation algorithms for this problem. Finally, Sect. 7 concludes the paper.

2. Structural Robustness against Delay Variation

In this paper, we study the application specific register-transfer level datapath design considering delay variations. As the first attempt for it, an input application algorithm is assumed to be the set of operations and data-dependencies between operations, and is assumed to be described as a data flow graph (DFG). Throughout this paper, a, a', b, b', etc., represent data, and $O_a, O_{a'}, O_b, O_{b'}$, etc., represent operations, where $a, a', b, b', \ldots$ are the results of $O_a, O_{a'}, O_b, O_{b'}, \ldots$, respectively.

2.1 Setup and Hold Constraints

Figure 1 illustrates the correct timing of control signals with respect to the execution of operation O_b. We assume that O_b

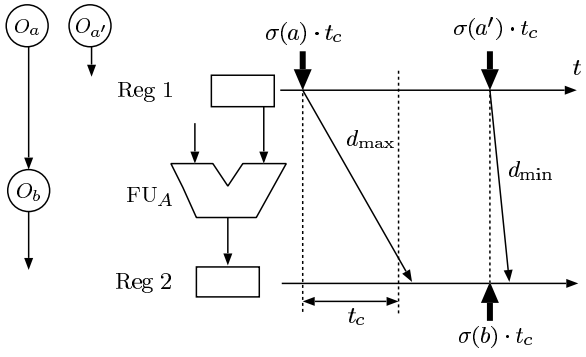


Fig. 1 Setup and hold constraints.

Manuscript received June 26, 2007.

[†]The authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi-shi, 923-1292 Japan.

a) E-mail: k-inoue@jaist.ac.jp

DOI: 10.1093/ietfec/e91–a.4.1044

is assigned to a functional unit FU_A, input data a for O_b is stored in a register Reg 1, and the result b of O_b is written to a register Reg 2. In this paper, we assume that a datapath is designed nominally under zero skew, i.e., the nominal delay from the clock source or a controller to the jth flip-flop of a register Reg i, $r(i, j)$, has the same value $r(i, j) = r_0$ for all i and j.

The arrival of the control signal at Reg 2 for latching data b has to be later than the arrival of b. This constraint is called "setup constraint" and is formulated as

$$\sigma(a) \cdot t_c + d_{\max} \leq \sigma(b) \cdot t_c,$$

where t_c is a clock period, $\sigma(x) \in \mathbb{Z}$ is the control step in which the controller sends out the control signal for latching data x, and $d_{\max}$ is the maximum path delay from Reg 1 to Reg 2 including the output delay of Reg 1 and the setup time of Reg 2. Note that, for simplicity in notation, the time axis is defined (the time origin is properly shifted) so that the arrival of a control signal (or a clock edge) at each register occurs nominally at a multiple of t_c, regardless of the value of r_0.

In general, a register is shared by several data. The control signal must arrive at Reg 2 to latch data b before b is destroyed. This constraint is called "hold constraint," and is formulated as

$$\sigma(b) \cdot t_c < \sigma(a') \cdot t_c + d_{\min},$$

where a' is the data that is written in Reg 1 after a, (that is, a is overwritten with a'), and $d_{\min}$ is the minimum path delay from Reg 1 to Reg 2 including the output delay of Reg 1 minus the hold time of Reg 2. A datapath operates correctly under no delay variations with the clock period t_c when the above two types of constraints are satisfied for each operation.

In this paper, we focus on two types of delay variations; one is the delay variation of the arrival of the clock signal (or a control signal) at a register, and the other is the delay variation of a path delay in the datapath. Sufficient conditions of the setup and hold constraints under delay variations can be formulated as

$$\begin{aligned}&\sigma(a) \cdot t_c + \Delta_{r(1)\max} + d_{\max} + \Delta_{d_{\max}} \\ &\quad \leq \sigma(b) \cdot t_c + \Delta_{r(2)\min}, \text{ and} \\ &\sigma(b) \cdot t_c + \Delta_{r(2)\max} \\ &\quad < \sigma(a') \cdot t_c + \Delta_{r(1)\min} + d_{\min} + \Delta_{d_{\min}},\end{aligned}$$

where $\Delta_{d_{\max}}$ and $\Delta_{d_{\min}}$ are variations of $d_{\max}$ and $d_{\min}$, respectively. In addition,

$$\begin{aligned}\Delta_{r(i)\min} &= \min_j\{\Delta_{r(i,j)}\}, \text{ and} \\ \Delta_{r(i)\max} &= \max_j\{\Delta_{r(i,j)}\},\end{aligned}$$

where $\Delta_{r(i,j)}$ is the delay variation of $r(i, j)$.

2.2 Structural Robustness against Delay Variation

A datapath operates correctly with a clock period t_c under a certain delay variation model if the setup and hold constraints considering delay variations are satisfied for each operation. But it would be difficult to estimate delays and delay variations correctly at the high level design. Therefore, we temporarily divide datapath properties into two categories; one is the functional property that is determined by a datapath structure and a control schedule, and the other is the performance property that is determined by parameter values together with the functional property. We focus on the functional property of a datapath in this paper. When exact delay values and exact delay variations are unknown, it is hard to guarantee the correct operation without any sacrifices. Our proposal is to offer a clock period as the expense of guaranteeing the correct operation of a datapath. This concept is described formally as follows.

Structural Robustness against delay Variation (SRV):
A datapath is said to have SRV if there always exists a clock period with which the setup and hold constraints considering delay variations are satisfied for each operation under any finite (or bounded) delay variations.

Please note the following remarks.

1. A datapath which operates correctly under a nominal situation does not always have the SRV property. Hence the datapaths having the SRV property form a proper subclass of datapaths.
2. Sacrificing the clock period as the functional property does not mean that the SRV property results in an inferior performance in speed. In fact, in most cases our datapath having SRV has comparable or superior performance than a non-SRV datapath, when both of them are designed with the same design environment and are operated in the same field environment.

In the following of this section, the SRV property is shaped more concretely.

2.3 SRV-type I

The setup constraint considering delay variations can be rewritten as

$$\begin{aligned}\Delta_{r(1)\max} + d_{\max} + \Delta_{d_{\max}} - \Delta_{r(2)\min} \\ \leq (\sigma(b) - \sigma(a)) \cdot t_c.\end{aligned}$$

Hence, we can have the condition for the existence of a clock period to satisfy the setup constraint under any finite delay variations as follows.

C1 : If a is one of the input data of O_b, then $\sigma(b) - \sigma(a) \geq 1$.

Of course, in a practical design, the minimum of $\sigma(b) - \sigma(a)$ will be specified with regarding practical values of the clock period t_c, delay $d_{\max}$ (and delay variations $\Delta_{d_{\max}}, \Delta_{r(1)\max}, \Delta_{r(2)\min}$, if they are properly estimated) and timing margin, and it can be more than one. Condition C1 is the minimum requirement with respect to the setup constraint for a datapath to have the SRV property.

On the other hand, the hold constraint considering delay variations can be rewritten as

$$(\sigma(b) - \sigma(a')) \cdot t_c < d_{\min} + \Delta_{d_{\min}} + \Delta_{r(1)\min} - \Delta_{r(2)\max}. \quad (1)$$

From this inequality, the condition for the existence of a clock period to satisfy the hold constraint under any finite delay variations is derived as follows.

C2 : If the input data of O_b is overwritten by another data a', then $\sigma(b) - \sigma(a') \leq -1$.

If a datapath satisfies both conditions C1 and C2, it has the SRV property. In particular, we call it SRV-type I.

2.4 SRV-type II

If the arrival time difference of control signals between bits in a register is properly small, the condition for the existence of a clock period to satisfy the hold constraint can be relaxed as follows.

C2'-1 : If the input data of O_b is overwritten by $a' \neq b$, then $\sigma(b) - \sigma(a') \leq -1$. In addition,
C2'-2 : the input data of O_b is allowed to be overwritten by the result b of O_b even through $\sigma(b) - \sigma(a') = \sigma(b) - \sigma(b) = 0$.

In case the input data of O_b is overwritten with b, Reg 2 (the output register for O_b) and Reg 1 (one of the input registers for O_b) are the identical one, and hence $\Delta_{r(2)\max} = \Delta_{r(1)\max}$ in inequality (1). The detailed assumption for C2'-2 is

$$d_{\min} + \Delta_{d_{\min}} + \Delta_{r(1)\min} - \Delta_{r(1)\max} > 0. \quad (2)$$

Remark that $\Delta_{r(1)\min}$ and $\Delta_{r(1)\max}$ in the above are the variations of the earliest arrival time and the latest arrival time among bits in a register, respectively, from a nominal arrival time of the same control signal (or clock edge).

Under the assumption (2), if a datapath satisfies conditions C1, C2'-1, and C2'-2, it has the SRV property. In particular, we call it SRV-type II.

2.5 SRV-type III

In the inequality (1), if $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is guaranteed, $d_{\min} + \Delta_{d_{\min}} + \Delta_{r(1)\min} - \Delta_{r(2)\max} > 0$ is always satisfied. Therefore, the condition for the existence of a clock period to satisfy the hold constraint can be reduced to

$$\sigma(b) - \sigma(a') \leq 0.$$

Hence, condition C2'-1 can be relaxed to the following C2'-1'.

C2'-1' : When the input data of O_b is overwritten by $a' \neq b$, if $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is guaranteed, then $\sigma(b) - \sigma(a') \leq 0$, otherwise $\sigma(b) - \sigma(a') \leq -1$.

If a datapath satisfies C1, C2'-1' and C2'-2 with a feasible order assignment of arrival timing of control signals (or clock edges) to registers, it has the SRV property. We especially call it SRV-type III. In a practical design, in order to guarantee $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ (i.e. a control signal (or a clock edge) arrives at Reg 2 earlier than Reg 1), we need to pay special cares on the order of arrival timing throughout the following design stages, such as logic design and layout design, but it is beyond the scope of this paper.

3. Register Assignment for SRV

The conditions for the SRV property relate directly to a register assignment. This section describes conditions of register sharing for a datapath to have the SRV property.

3.1 Condition of Register Sharing for SRV-type I

Basically multiple data can share a single register when their lifetimes do not overlap each other, where the lifetime of data a is defined conventionally to begin at $(\sigma(a) \cdot t_c)_+$ and to end at

$$\left(\max_{\substack{\text{for operation } O_b \text{ which} \\ \text{uses data } a \text{ as input}}} \{\sigma(b) \cdot t_c\} \right)_-,$$

and $(\tau)_+$ and $(\tau)_-$ represent "immediately after" τ and "just before" τ, respectively. However, the condition C2 requests

$$\sigma(b) - \sigma(a') \leq -1$$

when an input data a for O_b is overwritten with a', i.e., a' shares the same register with a. It means that we cannot assign a data which begins at $(\sigma(a') \cdot t_c)_+$ to the same register with a data which ends at $(\sigma(a') \cdot t_c)_-$, but we can do it with a data which ends at $((\sigma(a') - 1) \cdot t_c)_-$ or before (Fig. 2).

3.2 Conditions of Register Sharing for SRV-type II

For SRV-type II, the sharing feasibility is separated into two types depending on the relation between data. If a data a' is not the result of the latest operation O_b among the operations

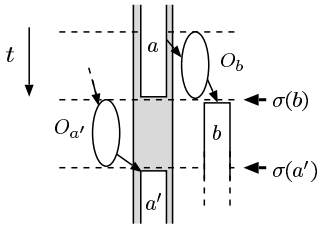


Fig. 2 Feasible register sharing. A rectangle represents a lifetime of a data, and an oval represents a scheduled operation. A gray belt corresponds to a register, and a data lifetime located on the belt means that the data is assignment to the register corresponding to the belt. The condition C2 (and C2'-1) means that, if O_b is the last operation that uses data a as an input, we cannot assign any other data (data b is excepted in C2'-1) soon after the end of the lifetime of a, but we can assign a data if its lifetime begins at one control step or more later than the end of a.

that use data a as their inputs, the sharing feasibility between a' and a is the same with the one for SRV-type I. That is, we cannot assign a' which begins at $(\sigma(a') \cdot t_c)_+$ to the same register with data a which ends at $(\sigma(a') \cdot t_c)_-$. We can do it with a data which ends at $((\sigma(a')-1) \cdot t_c)_-$ or before (Fig. 2).

On the other hand, if O_b is the sole latest operation among the operations that use data a as their inputs, we can assign the result b of O_b to the same register with the data a even though b begins at $(\sigma(b) \cdot t_c)_+$ and a ends at $(\sigma(b) \cdot t_c)_-$ (Fig. 3(a)). However, if there is more than one latest operation which uses data a as their input, none of their results can be assigned to the same register with a as shown in Fig. 3(b). This is because, when we assign b to the same register with data a based on C2'-2 with respect to the operation O_b, we need to apply C2'-1 with respect to the operation $O_{b'}$ (that is, in C2'-1, $O_{b'}$ is substituted for O_b, b' for b, b for a', and $b \neq b'$ for $a \neq b$), and it requests

$$\sigma(b') - \sigma(b) \leq -1$$

which contradicts with the assumption.

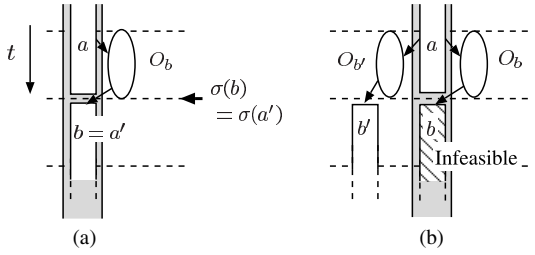


Fig. 3 Feasible register sharing. (a) If O_b is the last operation that uses data a as an input, we can assign b to the same register with a, however (b) if there are more than one last operations with respect to data a, none of their results can be assigned to the same register with a.

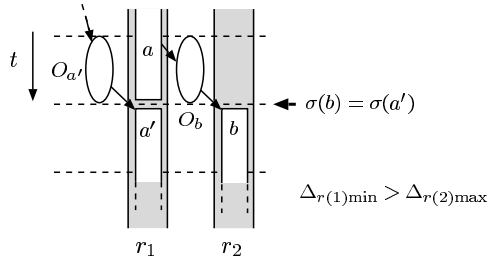


Fig. 4 Feasible register sharing for SRV-type III. Assume that b is the result of the last operation that uses data a as an input, and data a and b are assigned to the registers r_1 and r_2, respectively. If $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is guaranteed, any data a' that begins soon after the end of a can be assigned to the same register with a (i.e. r_1).

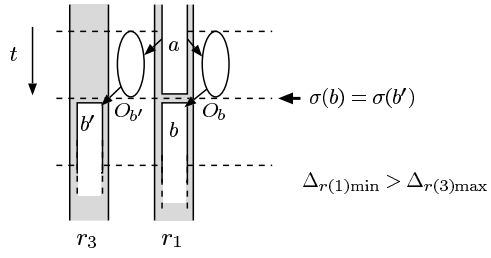


Fig. 5 Feasible register sharing for SRV-type III. Assume that data b and b' are results of the two latest operations that use data a as an input, and r_1 and r_3 are registers to which a and b' are assigned, respectively. If $\Delta_{r(1)\min} > \Delta_{r(3)\max}$ is guaranteed, data b can be assigned to the same register with a (i.e. r_1).

3.3 Register Sharing for SRV-type III

If a partial order among arrival timing of control signals (or clock edges) at registers exists, the sharing feasibility for SRV can be relaxed as follows. That is, if $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is guaranteed, the condition C2'-1 can be relaxed to the condition C2'-1';

$$\sigma(b) - \sigma(a') \leq 0.$$

It means that we can assign a data which begins at $(\sigma(a') \cdot t_c)_+$ to the same register with a which ends at $(\sigma(a') \cdot t_c)_-$ (Fig. 4). Moreover, in case there is more than one latest operation which uses data a as an input, we can assign one of these results to the same register with a if the arrival timing of a control signal (or clock edges) at the register to which a is assigned is later than the other registers (Fig. 5). In case the order of the arrival timing of control signals between two registers does not meet the prerequisite condition, the sharing feasibility is the same with the one for SRV-type II.

4. Comparison to Existing Techniques

The hold constraint under delay variations is the key factor of our SRV property and our feasible register assignment for SRV. In this section, two existing techniques to ensure the hold constraint are exhibited, and the position of our SRV in relation to those techniques is presented.

4.1 Register Configurations

As far as we know, there are two types of existing register configurations that can ensure the hold constraints under delay variations.

A "double-latch" register is a register configuration which connects two latches in series and uses two non-overlapping clock signals ϕ_1 and ϕ_2 as shown in Fig. 6(a). Timing behavior of the double-latch register is shown in

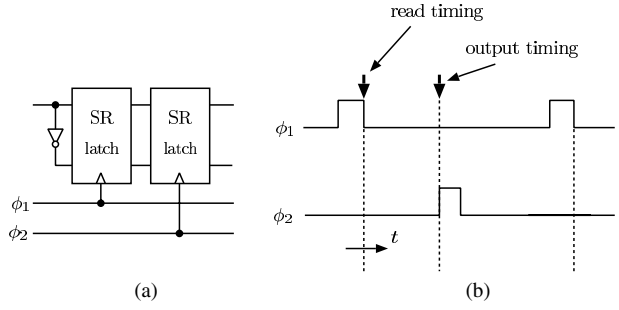


Fig. 6 The structure of a double-latch register (a) and its timing behavior (b).

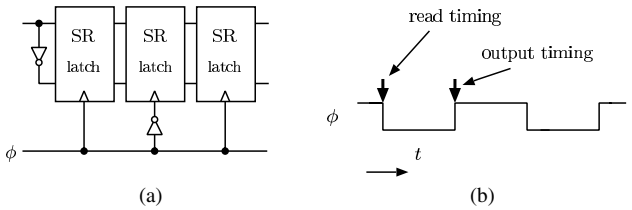


Fig. 7 The structure of a double-slave register (a) and its timing behavior (b).

Fig. 6(b). Problems of the double-latch register are the area overhead caused by using two clock signals and the increase in power consumption. In addition, it is necessary to keep the margin, i.e., difference between falling clock edge of ϕ_1 and rising clock edge of ϕ_2, large enough to absorb clock skew, clock jitter, and delay variations. Furthermore, this margin is a completely wasted time for computations. The larger delay variation becomes, the larger the margin has to be allocated. As a result, the fraction of the time that can be used for computations (in the following, we call it effective computation time) becomes smaller as shown in Fig. 8(a).

On the other hand, a "double-slave" register is a register configuration which connects three latches in series and uses one clock signal ϕ as shown in Fig. 7(a). Timing behavior of the double-slave register is shown in Fig. 7(b). While the double-slave register uses only one clock signal, it has the area overhead caused by using three latches. In addition, similarly to the double-latch register, the larger the margin is allocated, the smaller the fraction of the effective computation time becomes as shown in Fig. 8(b).

Compared with those register configurations, our SRV-based register assignment has superiority in timing issue. This is because,

(1) we do not need to modify a given operation schedule (we can use the same operation schedule with a double-latch-based or double-slave-based datapath configuration),
(2) our configuration can make use of the whole time of a clock period for data propagation from register to register including computation on a functional unit, while double-latch registers and double-slave registers can use only a part of a clock period left after taking an appropriate margin (we can always choose a shorter clock period than a double-latch-based or double-slave-based datapath configuration),
(3) in addition, our datapath guarantees a large timing margin almost or more than one clock period without sacrificing effective computation time (Fig. 8(c)).

With respect to the power (dynamic power without considering 0-1 correlation between data), our configuration does not increase the number of transitions of gate-outputs compared to a nominal configuration, while the double-latch-based datapath configuration doubles clock distribution power and the double-slave-based datapath configuration consumes 1.5 times larger register power.

Furthermore, the robustness of our datapath against delay variation does not depend on the choice of the register configuration, and we can always find an available clock frequency for the datapath with SRV to operate correctly under any finite (or bounded) delay variations.

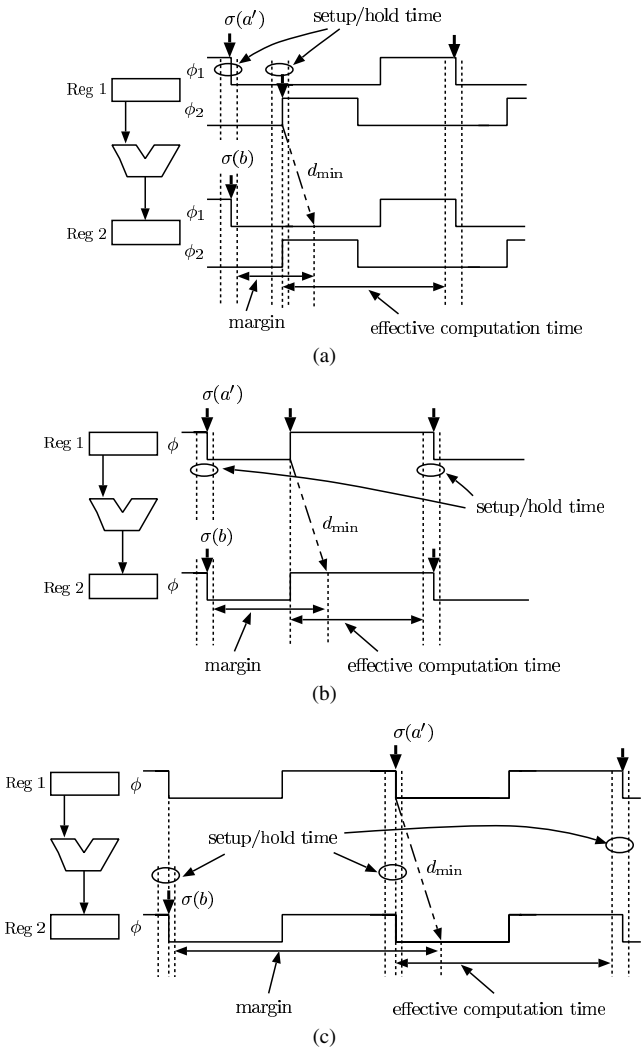


Fig. 8 Timing behavior of each register configurations. (a) A double-latch register, (b) a double-slave register, and (c) proposed SRV-based register assignment.

4.2 Compensation of the Minimum Path Delay

Another possible approach to ensure the hold constraint is to enlarge the minimum path delay between registers. It can be done by inserting delay elements on non-critical paths mainly in a functional unit. In the following, we call it "minimum delay compensation (MDC)." MDC and our SRV (or SRV-based register assignment) are compatible techniques, and we show this using an example in the next section.

5. Applications

In this section, we demonstrate several design examples based on our SRV and SRV-based register assignment. As

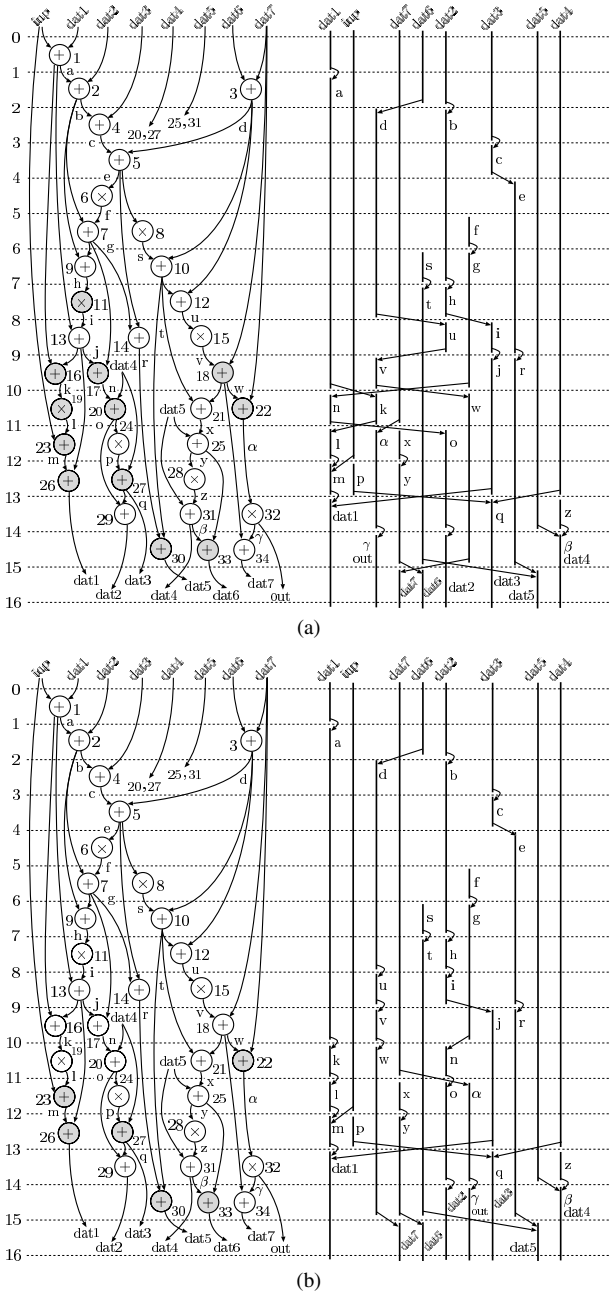


Fig. 9 Schedule and register assignment. (a) Shows one design example which is obtained without considering the SRV property, and (b) shows another design whose MDC is minimized by applying SRV-based register assignment.

an input application, we use the fifth-order wave digital elliptic filter [9]. First we design a schedule as shown in Fig. 9, and use this schedule in common for all designs demonstrated in this section. Note that the schedule is designed with assuming that three adders and one multiplier are available and every operation is a single-cycle operation. The time chart shown in Fig. 9(a) (Fig. 9(b), Fig. 10(a), and Fig. 10(b) also) consists of two parts. The left half part (schedule table) shows operation schedule together with dependency arcs. Each operation is identified by a numeric and each data is identified by an alphabet. The symbol "inp" and "out" represent the primary input and primary output data, respectively, and a symbol "dat i" represents a inter-iteration data. On the other hand, the right half part (register assignment table) shows data lifetimes with register assignment information. That is, each vertical line segment represents the lifetime of one data, and line segments arranged in the same column mean that the corresponding data are assigned to the same register. A directed edge from a lifetime L_i to L_j means that L_j is the output of the operation that uses L_i lastly.

Now, we recall our stance on approaching delay variation aware datapath design. We have separated the structural property from the performance property, where the former is mainly determined by the topological structure of the datapath and the schedule in control steps and the latter is determined by both the structural property and particular parameter values. The proposed SRV relates the former and guarantees that, under any delay variations, our datapath always operates correctly by only adjusting the clock period. Because of this stance, in the following demonstrations, datapaths are designed with focusing mainly on the structural property without specifying a particular technology nor a particular delay variation. Datapath optimization in performances under a particular technology and a particular delay variation is a future problem.

5.1 Combination of Our SRV-Based Register Assignment with MDC

In this section, we demonstrate the possibility that the number of functional units which require MDC can be decreased by applying our SRV-based register assignment. Figure 9(a) shows one possible minimum register assignment. Gray operations in the schedule table are operations which require MDC for assuring the hold constraint. As a result, all of three adders and one multiplier require MDC.

On the other hand, Fig. 9(b) shows another minimum register assignment. The maximum degree of temporal overlap of gray operations is now two, and it is enough to apply MDC to two adders. The reduction of the number of functional units which require MDC is achieved by applying the SRV-based register assignment as much as possible while keeping the total number of registers unchanged. That is, if conditions C2'-1 and C2'-2 are satisfied for an operation O_b, then we do not need to apply MDC to the functional unit that executes O_b. On the other hand, we need to apply MDC to an operation O_b (strictly speaking, the functional unit which executes O_b) for which C2'-1 is not satisfied.

This observation motivates us to formulate an optimization problem to minimize the number of functional units which require MDC with keeping the number of registers unchanged, and it will be tackled in future.

5.2 Complete SRV Designs

Figure 10(a) shows a complete SRV design (register assign-

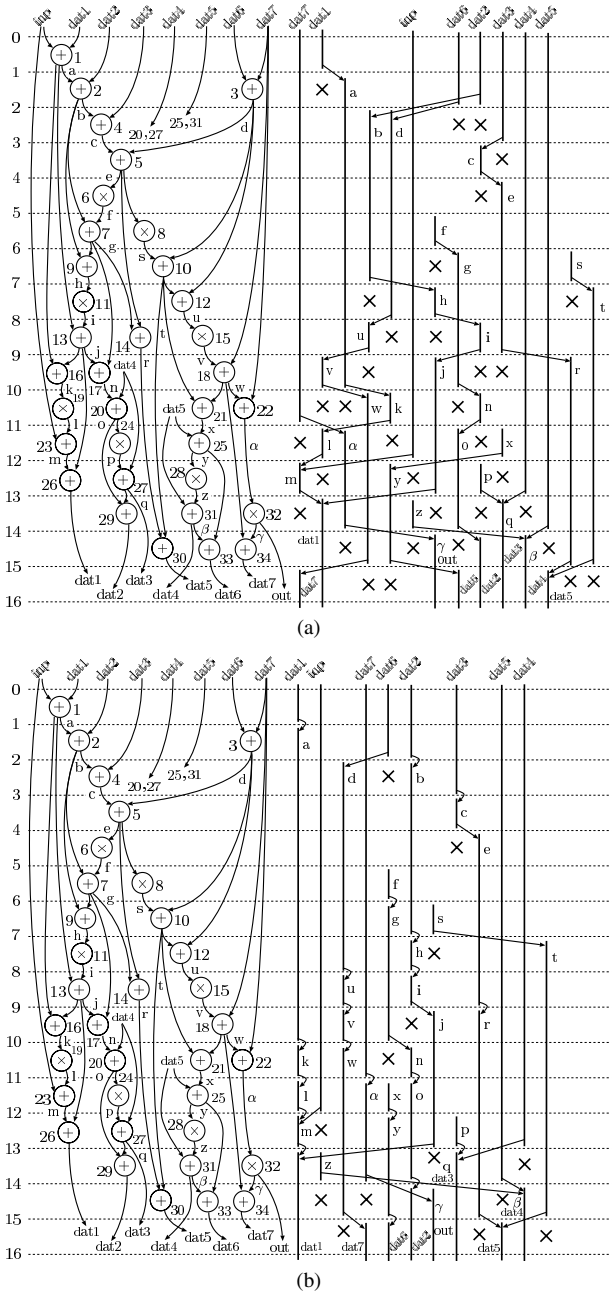


Fig. 10 Schedule and register assignment. Symbol "×" represents a control step to which any data cannot be assigned. (a) Shows the resultant register assignment based on the conditions for SRV-type I, and (b) shows the one based on the conditions for SRV-type II.

ment) based on SRV-type I, and Fig. 10(b) shows another complete SRV design based on SRV-type II. In these designs, the hold constraints are all resolved by our SRV-based register assignment, and neither MDC nor specially configured registers are needed. As a result, these datapaths have the SRV property and also have a large margin more than one clock cycle on the hold constraint. However the number of registers might be increased.

The number of registers required for our feasible register assignment based on SRV is at least the maximum of the number of intersecting symbols "×" and data lifetimes in each control step in Fig. 10. Actually, compared with the conventional minimum register assignment, three more registers are needed for SRV-type I (Fig. 10(a)), and one more register for SRV-type II (Fig. 10(b)). For those instances, the lower bound on the number of registers given above has been achieved. If we use double-latch registers or double-slave registers, we can adopt the conventional minimum register assignment, and can save three registers and one register compared to Fig. 10(a) and Fig. 10(b), respectively. However, as it is mentioned in the previous section, a double-latch-based datapath has the area overhead caused by using an additional clock signal, and a double-slave-based datapath has the area overhead caused by using an additional second slave. Our increase in the number of registers may possibly be comparable to those overheads in double-latch-based datapaths and double-slave-based datapaths. Detailed comparison may depend deeply on a target application and a target technology, and is beyond the scope of this paper.

In this paper, we assume that a schedule is given as a part of a problem instance. Under this situation, we have shown that the minimum register assignment for complete SRV-based designs type I and II are solved in polynomial time complexity by an extended left-edge algorithm. When a DFG includes directed cycles, we also have shown that the complexity of the minimum register assignment problem is $\mathcal{NP}$-hard [10]. On the other hand, the minimum register assignment for the SRV-based design type III is still an open problem.

6. Minimum Register Assignment for SRV

The register assignments demonstrated in Fig. 10 are designed as the one for a loop pipeline application. Unfortunately, the minimum register assignment problem (find a register assignment which achieves the minimum number of registers) for loop pipeline applications based on SRV is concluded to be $\mathcal{NP}$-hard, as its conventional version is $\mathcal{NP}$-hard. The development of an efficient heuristic algorithm of the minimum register assignment for loop pipeline applications remains as a future problem.

As for an application which is represented by an acyclic DFG (a DAG application, in short), we have the following results.

Theorem 1: The minimum register assignment problem for DAG applications based on SRV-type I is in the class $\mathcal{P}$.

Theorem 2: The minimum register assignment problem for DAG applications based on SRV-type II is in the class $\mathcal{P}$.

As for the minimum register assignment problem for DAG applications based on SRV-type III, its computational complexity as well as a solution algorithm are still open problems.

In the following of this section, we show register assignment algorithms for SRV-type I and II in case an input DFG is a directed acyclic graph.

6.1 Minimum Register Assignment for SRV-type I

From the condition C2, any data cannot be assigned to a register soon after the end of another data assigned to the same register. This situation can be simulated by extending each data lifetime. After the extension, our problem is reduced to the conventional register assignment problem.

Register assignment algorithm for SRV-type I

Step 1. Extend each data lifetime by one control step from the last step.

Step 2. Apply the left-edge algorithm [11] to the set of the extended lifetimes obtained in Step 1.

Theorem 3: The proposed algorithm for SRV-type I computes a minimum register assignment which satisfies SRV-type I.

As for DAG applications, the register assignment based on SRV-type I, as well as the conventional register assignment, can be reduced to the coloring problem for interval graphs, which has been proven to be solvable in polynomial time. The left-edge algorithm is a well-known method to solve this problem, and it can be seen in many literatures. Algorithm details and a proof of the optimality of the solution are skipped in this paper.

6.2 Minimum Register Assignment for SRV-type II

The following shows the proposed algorithm for register assignment considering the conditions C2'-1 and C2'-2.

Register assignment algorithm for SRV-type II

Step 1. Find a pair of data where one is the result of the sole latest operation that uses the other data as the input (please refer to the condition C2'-2), and decide to assign those data to the same register. As a result of this decision, merge their lifetimes (intervals) to a single lifetime (a single interval). Repeat the above procedure as much as possible.

Step 2. Add one control step to the last step of the intervals obtained in Step 1 and the lifetimes without being merged, and they are called extended lifetimes.

Step 3. Apply the left-edge algorithm to the set of the extended lifetimes obtained in Step 2.

Theorem 4: The proposed algorithm for SRV-type II computes a minimum register assignment which satisfies SRV-type II.

In the following discussion, a register assignment is a mapping $A : \mathcal{D} \to \mathcal{R}$, where $\mathcal{D}$ is a set of data, $\mathcal{R}$ is a set of positive integers which represent register IDs. We assume that the data are assigned to registers in ascending order of register IDs by our proposed algorithm. Now we let $\ell_s(d)$ and $\ell_e(d)$ be the start and the end control steps of lifetime of data d, respectively (Fig. 11). $\mathcal{R}_A(d)$ is a set of data which are assigned to the same register with d and their start control steps are later than or equal to $\ell_s(d)$ in A.

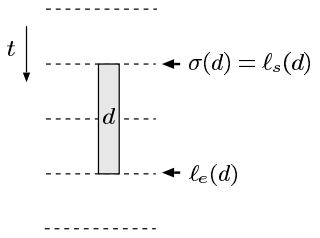


Fig. 11 The start and the end control steps of the lifetime of d.

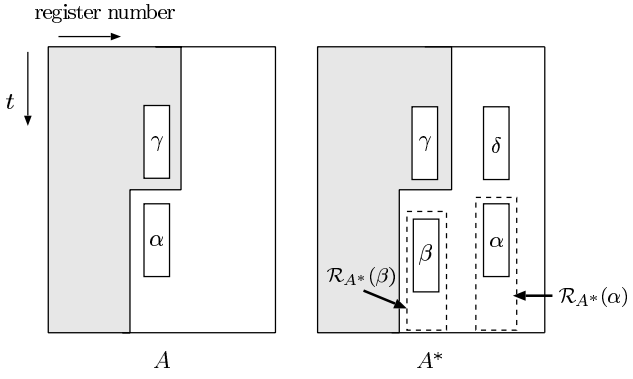


Fig. 12 Register assignment A and A^*, where α, β, δ and γ are data. A shaded region contains data which are assigned to registers identically in both A and A^*.

Proof of Theorem 4: Let A be a register assignment obtained by our proposed algorithm, and let A^* be a minimum register assignment for SRV-type II. We compare A^* with A in the manner of one data to another with the same order in which our left-edge algorithm in Step 3 chooses lifetimes. If the lifetime is the composite of several lifetimes formed in Step 2, the order follows the order of their start control steps (earliest one first). Since $A^* \neq A$, we can find the last data γ such that data from the beginning to γ are assigned identically in both A and A^*, but the next to γ in A is data α and the earliest data which starts after γ and is assigned to the same register with γ is $\beta, \beta \neq \alpha$, in A^*. Let δ be the latest data assigned before α in A^* (Fig. 12). Note that, if there are multiple minimum register assignments, we choose such A^* that it has the maximum number of identically-assigned data from the beginning to γ.

Now, we represent the situations of A and A^* by using a triple (x, y, z), where x, y and z are Boolean variables and their values are defined as follows.

x : If $\ell_e(\gamma) = \ell_s(\alpha)$ in A then 0, otherwise 1.

y : If $\ell_e(\gamma) = \ell_s(\beta)$ in A^* then 0, otherwise 1.

z : If $\ell_e(\delta) = \ell_s(\alpha)$ in A^* then 0, otherwise 1.

In case $(x, y, z) = (0, 0, 0), (0, 0, 1)$: The fact that γ and α can be assigned to the same register and $\ell_e(\gamma) = \ell_s(\alpha)$ means that O_α is the sole latest operation that uses γ as the input. Similarly, the fact that γ and β can be assigned to the same register and $\ell_s(\gamma) = \ell_s(\beta)$ means that O_β is the sole latest operation that uses γ as the input. Those two situations never happen simultaneously.

In case $(x, y, z) = (0, 1, 0), (0, 1, 1), (1, 1, 1)$: We can swap register numbers assigned to data in $\mathcal{R}_{A^*}(\alpha)$ for register numbers assigned to data in $\mathcal{R}_{A^*}(\beta)$. The resultant register assignment A^{**} satisfies conditions for SRV-type II and achieves the same number of registers with A^*. As a result, A^{**} is a minimum register assignment and has one more data in common with A than A^*. It is a contradiction to the choice of A^*.

In case $(x, y, z) = (1, 0, 0), (1, 1, 0)$: The fact that δ and α can be assigned to the same register and $\ell_e(\delta) = \ell_s(\alpha)$ means that O_α is the sole latest operation that uses δ as an input. In this situation, our proposed algorithm always finds the pair δ and α in Step 2, and decide to assign δ and α to the same register. Thus this situation never happen.

In case $(x, y, z) = (1, 0, 1)$: The fact that γ and β can be assigned to the same register and $\ell_e(\gamma) = \ell_s(\beta)$ means that O_β is the sole latest operation that uses γ as an input. If only γ is the input to O_β, our proposed algorithm always finds the pair γ and β in Step 2, and assigns γ and β to the same register. Our proposed algorithm does not assign γ and β to the same register only if O_β has another input $c \neq \gamma$ and O_β is also the sole latest operation that uses c as the input, and our algorithm assigns c and β to the same register. If $c = \delta$, swap register numbers assigned to data in $\mathcal{R}_{A^*}(\beta)$ for register numbers assigned to data in $\mathcal{R}_{A^*}(\alpha)$. If $c \neq \delta$, let e be a data assigned to immediately after c. Swap register numbers assigned to data in $\mathcal{R}_{A^*}(\beta)$ for register numbers assigned to data in $\mathcal{R}_{A^*}(e)$, and swap register numbers assigned to data in $\mathcal{R}_{A^*}(e)$ for register numbers assigned to data in $\mathcal{R}_{A^*}(\alpha)$. Let A^{**} be the register assignment obtained by this operation. A^{**} is again a register assignment which satisfies conditions for SRV-type II and achieves the minimum number of registers, and it has one more data in common with A than A^*. It is a contradiction to the choice of A^*. ■

7. Conclusions

We have proposed the concept of the structural robustness against delay variation (SRV), and we have presented sufficient conditions for a datapath to have SRV. Compared with conventional designs, a resultant circuit designed under those conditions has a large timing margin to delay variations without sacrificing effective computation time. In addition, under any degree of delay variations, we can always find an available clock frequency for a resultant datapath to operate correctly. The latter characteristic might be useful especially for IP-based designs, since those IPs are reused in various design environments, in various applications, and in various field environments, and it might be important for IPs to guarantee the correct operation in such various environments.

Discussions done in this paper relate to structural properties, and datapath optimization in performances under a particular technology and a particular delay variation is one of major future problems.

Acknowledgments

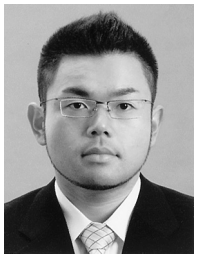
The authors would like to thank the anonymous reviewers for their useful suggestions. This work is partly supported by the Society for the Promotion of Science, Japan, under Grant-in-Aid for Scientific Research (C) No. 19560340, 2007–2008.

References

[1] R. Kumar and V. Fursun, "Impact of temperature fluctuations on circuit characteristics in 180 nm and 65 nm CMOS technologies," Proc. 39th International Symposium on Circuits and Systems, pp.21–24, May 2006.
[2] M. Hashimoto, J. Yamaguchi, T. Sato, and H. Onodera, "Timing analysis considering temporal supply voltage fluctuation," Proc. 10th Asia South Pacific Design Automation Conference, pp.1098–1011, Jan. 2005.
[3] P. Heydari and M. Pedram, "Analysis and reduction of capacitive coupling noise in high-speed VLSI circuits," Proc. 19th International Conference on Computer Design, pp.104–109, Sept. 2001.
[4] M. Conti, G.-F.D. Betta, S. Orcioni, G. Soncini, C. Turchetti, and N. Zorti, "Test structure for mismatch characterization of MOS transistors insubthreshold regime," Proc. IEEE International Conference on Microelectronic Test Structures, vol.10, pp.173–178, 1997.
[5] S. Matsumoto, H.J. Mattausch, S. Ooshiro, Y. Tatsumi, M. Miura-Mattausch, S. Kumashiro, T. Yamaguchi, K. Yamashita, and N. Nakayama, "Test-circuit-based extraction of inter- and intra-chip MOSFET-performance variations for analog-design reliability," Proc. IEEE Custom Integrated Circuits Conference, pp.582–585, 2001.
[6] K. Nose, M. Kajita, and M. Mizuno, "A 1 ps-resolution jitter-measurement macro using interpolated jitter oversampling," Proc. International Solid-State Circuits Conference, pp.2112–2121, Feb. 2006.
[7] M. Murakawa, E. Takahashi, T. Susa, and T. Higuchi, "Post-fabrication clock timing adjustment for digital LSIs with generic algorithms ensuring timing margins," Report of MIRAI Project, 2004.
[8] J. Xi and W.W.-M. Dai, "Jitter-tolerant clock routing in two-phase synchronous systems," Proc. 1996 IEEE/ACM International Conference on Computer-Aided Design, pp.316–320, 1996.
[9] P. Michel, U. Lauther, and P. Duzy, The Synthesis Approach to Digital System Design, Kluwer Academic Publishers, 1992.
[10] K. Inoue, Structural delay-variation tolerance and high level synthesis for datapaths, Master Thesis, Japan Advanced Institute of Science and Technology, 2007.
[11] F.J. Kurdahi and A.C. Parker, "REAL: A program for register allocation," Proc. 24th Design Automation Conference, pp.210–215, 1987.

Keisuke Inoue received the B.E. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2005, and the M.E. degrees in information science from Japan Advanced Institute of Science and Technology, Ishikawa, Japan, in 2007. He is currently D.E. student of Information Science in Japan Advanced Institute of Science and Technology. His research interests are in VLSI design automation.

Mineo Kaneko received Bachelor of Engineering, Master of Engineering, and Doctor of Engineering degrees in electrical and electronic engineering from Tokyo Institute of Technology in 1981, 1983, and 1986, respectively. From 1986 to 1996, he worked at Tokyo Institute of Technology as a research associate, a lecturer, and an associate professor. In 1996, he transferred to Japan Advanced Institute of Science and Technology (JAIST), and currently he is a professor in the Graduate School of Information Science, JAIST. His research interests include circuit theory, CAD for VLSIs, and signal processing. He is a member of IEEE and ACM.



Tsuyoshi Iwagaki received the B.E. degree in electronic engineering from Osaka Institute of Technology, Osaka, Japan, in 2000, and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan, in 2002 and 2004 respectively. Presently, he is an assistant professor at the School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan. His research interests include VLSI CAD, design for testability and test generation. He received the fourth IEEE Workshop on RTL and High Level Testing Best Paper Award in 2004, and IEEE Kansai Section Student Paper Award in 2005. He is a member of IEEE and IPSJ.