Minimizing Minimum Delay Compensations for Timing Variation-Aware Datapath Synthesis

Keisuke Inoue† Mineo Kaneko† Tsuyoshi Iwagaki†
†School of Information Science, Japan Advanced Institute of Science and Technology (JAIST)
1-1, Asahidai, Nomi-shi, Ishikawa 923-1292 JAPAN Email: {k-inoue, mkaneko, iwagaki}@jaist.ac.jp

Abstract— **As the feature size of transistors becomes smaller, delay variations become a serious problem in VLSI design. In many cases, the hold constraint, as well as the setup constraint, becomes critical for latching a correct signal under delay variations. One approach to ensure the hold constraint under delay variations is to enlarge the minimum-path delay between registers, which is called minimum-path delay compensation (MDC) in this paper. MDC can be done by inserting delay elements mainly in non-critical paths of a functional unit (FU). This paper is the first attempt to discuss an optimization problem to minimize the number of FUs which require MDC in datapath synthesis. One of our contributions is to show that the problem is NP-hard in general, and it is in the class P if the number of FUs is a constant. In addition, a polynomial time algorithm for the latter is another contribution. The proposed method generates a datapath having (1) robustness against delay variations, which is ensured partly by MDC technique and partly by SRV-based register assignment, and (2) the minimum possible numbers of MDCs and registers.**

I. Introduction

With the advance of process technologies, the feature size of transistors in a VLSI becomes smaller, and switching delay becomes shorter. 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. There are several reports on this problem from different points of view. In order to reduce the timing margin, methods for estimating delay variations more precisely were studied in [1]. In [2], the authors addressed the problem to correct timing violations after manufacturing by using programmable delay elements (PDEs). Reference [3] proposed a performance yield constrained high-level synthesis based on a statistical static timing analysis (SSTA). Reference [4] introduced the concept "critical path isolation" for variation-tolerant datapaths. Most of those works target so-called "setup constraint" because it directly limits the maximum available clock frequency. On the other hand, so-called "hold constraint" does not affect directly the clock frequency, and less attention has been paid even though it is indispensable for the correct operation of a datapath.

Typical circuitry to ensure the hold constraint is the "double-latch" with two non-overlap clocks; one drives master latches and the other does slave latches [5]. The timing margin for the hold constraint is then set by a phase difference between these two clocks. However, it has two major drawbacks; one is the cost of delivering two clock signals, and the other is the erosion of the effective execution time (time from the output-change of a flip-flop to the input-latch of a flip-flop), which might result in a longer clock period in compensation for this erosion.

In this paper, two different approaches to ensure the hold constraint without degrading the setup timing margin are introduced, combined and optimized for designing a cost effective datapath having robustness against delay variations. Note that our design target is a register-transfer level datapath circuit, and hence a register-to-register path is a multiple-input, multiple-output combinational circuit. It means that a register-to-register path consists of more than one logic path. "The minimum- (maximum-) path delay from one register to another" is the minimum- (maximum-) logic path delay over all those logic paths between specified two registers.

We focus on two types of delay variations. One is the delay variation of the arrival of a control signal at a register, and the other is a register-to-register path delay variation in a datapath part. One approach to ensure the hold constraint is to enlarge the minimum path delay between registers. It can be done by inserting delay elements mainly in non-critical paths of a functional unit (FU). In the following, we call this technique the minimum-path delay compensation (MDC). MDC is a simple technique, but it has several drawbacks such as area overhead, extra power consumption. Therefore the number of FUs which need MDC should be minimized. Another approach to ensure the hold constraint under delay variation is a kind of constrained register assignment which is named structural robustness against delay variation (SRV)-based register assignment [6] [7], but it tends to need more registers than a conventional register assignment. This paper treats a novel problem to minimize FUs which need MDC, where such minimization will take place by incorporating SRV-based register assignment. A resultant datapath has (1) robustness against delay variations, which is ensured partly by MDC technique and partly by SRV-based register assignment, and (2) the minimum possible numbers of MDCs and registers.

II. Design Issue Considering Delay Variations

In this paper, we treat the register-transfer level datapath synthesis. An input application algorithm to the synthesis is assumed to be represented as a data flow graph (DFG), where vertices are operations, and arcs are data dependencies between operations. 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.

A. Setup and Hold Constraints

As briefly mentioned in Section I, a register-to-register path in our context is a multiple-input, multiple-output combinational circuit. Hence it consists of many logic paths, and the minimum- (maximum-) path delay from one register to another is the minimum- (maximum-) path delay among all those logic paths.

Fig. 1 illustrates the correct timing of control signals with respect to the execution of an operation O_b. We assume that O_b is assigned to a functional unit FU_A (we use ρ to represent FU assignment, like $\rho(O_b) = \mathrm{FU}_A$) , an 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 $r(i, j)$ from a clock source (or a controller) to the jth flip-flop (FF) of a register Reg i, has the same value $r(i, j) = r_0$ for all i and j. Note that a similar argument can be made for a datapath

978-1-4244-2167-1/08/$25.00 ©2008 IEEE

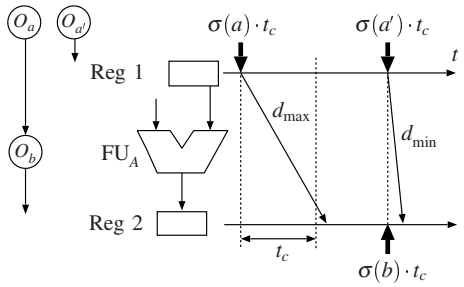

Fig. 1. Setup and hold constraints

designed with intentional skew, however we limit our discussion to a datapath designed with zero skew for the sake of simplicity.

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

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

where t_c is the 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 so that the arrival of a control signal at each register occurs nominally at a multiple of t_c.

In general, a register is shared by more than one data. If data a and a' share the same register Reg 1, and a is overwritten by a', the control signal for latching data b has to arrive at Reg 2 before b is destroyed. This is called "hold constraint" and is formulated as

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

where $d_{\min}$ 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.

In this paper, we focus on two types of delay variations. One is the variation of the arrival timing of a control signal at a register, and the other is the variation of a path delay in the datapath part. The following shows the setup and hold constraints considering delay variations,

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

where $\Delta_{d_{\max}}$ and $\Delta_{d_{\min}}$ are variations of $d_{\max}$ and $d_{\min}$, respectively. In addition, $\Delta_{r(i)\max} = \max_j\{\Delta_{r(i,j)}\}$ and $\Delta_{r(i)\min} = \min_j\{\Delta_{r(i,j)}\}$, where $\Delta_{r(i,j)}$ is the delay variation of $r(i,j)$.

B. Minimum-Path Delay Compensation (MDC)

One approach to ensure the hold constraint is to enlarge $d_{\min}$. It can be done by inserting delay elements mainly in non-critical paths of an FU. In the following, we call this technique the minimum-path delay compensation (MDC). In Fig. 1, the violation of hold constraint for writing b to a register Reg 2 is due to the increase of $-\Delta_{r(1)\min} + \Delta_{r(2)\max}$ in (1). If data a' which is written at the same timing with $\sigma(b)$ is assigned to Reg 1 ($\sigma(b) = \sigma(a')$), the hold constraint (1) can be reduced to

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

In this case, we need to apply MDC to FU$_A$, which increases $d_{\min}$ to $d_{\min} + d_{\text{MDC}}$, $d_{\text{MDC}} > 0$, and make $d_{\min} + d_{\text{MDC}}$ larger than $-\Delta_{r(1)\min} + \Delta_{r(2)\max} - \Delta_{d_{\min}}$ to satisfy (2) (Fig. 2.(a)). As a result, $d_{\min} + d_{\text{MDC}}$ is the improved margin for the hold constraint against the delay variation $-\Delta_{r(1)\min} + \Delta_{r(2)\max} - \Delta_{d_{\min}}$.

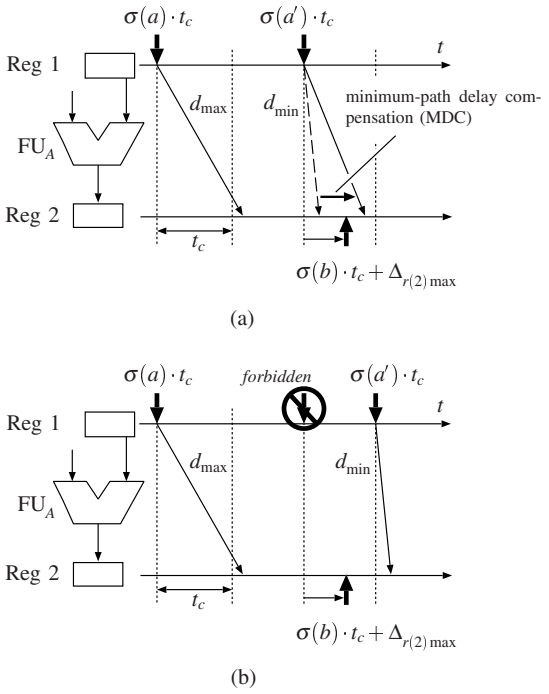

Fig. 2. Hold constraint in (a) is ensured by minimum-path delay compensation (MDC), on the other hand, hold constraint in (b) is ensured by disallowing to write data to the input register at the same timing to write output data (SRV-based register assignment).

C. SRV-based Register Assignment

In (1), $\sigma(a') = \sigma(b)$ endangers the hold constraint when delay variations are unavoidable. If we assign data x with $\sigma(x) = \sigma(b)$ to a different register with data a (an input to O_b), and we assign only data a' with $\sigma(a') \geq \sigma(b) + 1$ to the same register with data a (Fig. 2.(b)), we have

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

As a result, we can have a margin of more than one clock period for the hold constraint against the delay variation $-\Delta_{r(1)\min} + \Delta_{r(2)\max} - \Delta_{d_{\min}}$. Data assignment in which data x with $\sigma(x) = \sigma(b)$ is forbidden to be assigned to the same register with an input data to O_b is called SRV-based register assignment. Note that there is one exception. That is, when O_b is the sole operation that uses data a (an input to O_b) lastly, data b (the output of O_b) is allowed to be assigned to the same register with data a. (In this case, the control signal to overwrite a with $a' = b$ and the control signal to write b to a register Reg 2 are the same signal sent to the same register (Reg 1 = Reg 2). If we assume that the control timing difference between FFs in a register is negligible small ($\Delta_{r(1=2)\min} = \Delta_{r(2=1)\max}$) compared with $d_{\min}$, the hold constraint is reduced to $-\Delta_{r(1)\min} + \Delta_{r(2)\max} = 0 < d_{\min} + \Delta_{d_{\min}} + (\sigma(a' = b) - \sigma(b))\cdot t_c = d_{\min} + \Delta_{d_{\min}}$.)

III. MDC and SRV-Based Register Assignment

MDC is a simple technique, but it has several drawbacks such as area overhead, extra power consumption, degradation of the

maximum delay performance. SRV-based register assignment, on the other hand, tends to need more registers than a conventional register assignment. Our problem discussed in this paper is the combination of MDC with SRV-based register assignment. In the resultant datapath, each hold constraint of each operation is ensured by at least one of these two techniques. In this section, we formulate our problem and show a small demonstrative example.

A. Formulation

Our optimization problem, MDC and SRV-based register assignment problem receives (1) an application algorithm presented by a DFG $G = (\mathcal{O}, \mathcal{A})$, where $\mathcal{O}$ is a set of operations, $\mathcal{A}$ is a set of arcs, (2) a set of data $\mathcal{D}$, which has one-to-one correspondence to $\mathcal{O}$, (3) a control schedule $\sigma : \mathcal{O} \rightarrow \mathbb{Z}_+$, (4) a set of FUs $\mathcal{F}$ and a set of registers $\mathcal{R}$, and (5) FU assignment $\rho : \mathcal{O} \rightarrow \mathcal{F}$ as a problem instance, and finds a register assignment $\xi : \mathcal{D} \rightarrow \mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{F}$ of FUs that MDC is applied to so that every hold constraint in the application is ensured at least one of MDC technique or SRV-based register assignment, and $|\mathcal{C}|$ (the number of FUs that need MDC) is minimized.

If the hold constraint of an operation is ensured by SRV-based register assignment, we can have more than one clock margin for the operation, and if it is ensured by MDC technique, we can have a margin $d_{\min} + d_{\mathrm{MDC}}$.

B. Example

In this section, we explain our problem by using a small demonstrative example in Fig. 3.(a), where ovals are scheduled operations, rectangles are data lifetimes, and a directed edge (u, v) between lifetimes means that v is the output of the operation that uses u lastly. Assuming that two functional units FU_A and FU_B are available, FU assignment is done as $\rho(O_{a'}) = \rho(O_{a''}) = \mathrm{FU}_A$ and $\rho(O_b) = \mathrm{FU}_B$. Fig.s 3.(b) and 3.(c) are two feasible minimum register assignments.

In the first assignment Fig. 3.(b), data a in Reg 1 is overwritten by a' at the same timing that output b of O_b is written to Reg 2. If the arrival of control signal to write b to Reg 2 is later than the arrival of the fastest effect of the input change from a to a', incorrect data b polluted by this fastest effect may possibly be written to Reg 2. To avoid this malfunction, we need to apply MDC to FU_B. On the other hand, for the second register assignment in Fig. 3.(c), even if the arrival of control signal to write b to Reg 2 is delayed largely (within one clock period), correct result is written to Reg 2, because data a is not overwritten by any data at the timing that output b of O_b is written to Reg 2. It is corresponding to one clock period margin for hold constraint of O_b, and we do not need to apply MDC to FU_B. This example shows that the register assignment affects the necessity of MDC.

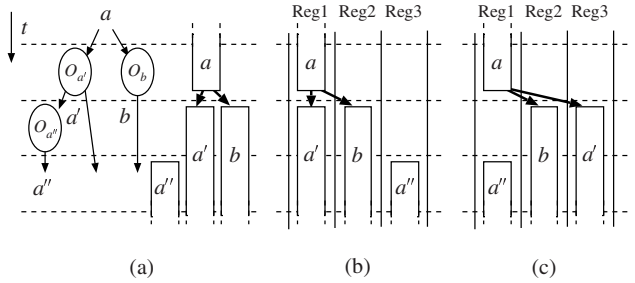

Fig. 3. A problem instance (a), and two feasible minimum register assignments (b) and (c).

C. Computational Complexities

If an input DFG is restricted to a directed acyclic graph (DAG), we have the following theorems.

Theorem 1: If the number of FUs is a variable, MDC and SRV-based register assignment problem is **NP**-hard.

Theorem 2: If the number of FUs is a fixed constant, MDC and SRV-based register assignment problem is in the class **P**.

The proof of Theorem 1 is based on a polynomial time reduction from SET_PACKING [8] to the decision version of our problem. The details of this proof [9] are omitted here for lack of space. On the other hand, Theorem 2 can be proven by showing a polynomial time computation algorithm for the problem. The next section is devoted to this polynomial time computation algorithm.

IV. Polynomial Time Algorithm

First, we describe the kernel subroutine of our algorithm, which receives a set of FUs $\mathcal{C}$ to which MDC is applied as well as the set of all the FUs $\mathcal{F}$, and outputs a minimum register assignment.

SRV-based register assignment for a given MDC assignment :

Step 1. Find a pair of data where one is the result of a sole operation that is executed on an FU in $\mathcal{F} - \mathcal{C}$ and uses the other data lastly as an input, and decide to assign these data to the same register. As a result of this decision, merge their lifetimes (intervals) to a single lifetime (a single interval). Repeat the procedure as much as possible.

Step 2. Let D be a set of all the intervals obtained in Step 1 and the other lifetimes that are not merged. In addition, Let $D' \subseteq D$ be a set of intervals each of which has an operation that uses the corresponding data lastly and is executed on an FU in $\mathcal{F} - \mathcal{C}$. Add one control step to the last step of each element in D'. The resultant lifetimes obtained in this step are called extended lifetimes.

Step 3. Apply the left-edge algorithm [10] to the set of the extended lifetimes obtained in Step 2. □

Lemma 1: SRV-based register assignment for a given MDC assignment computes a minimum register assignment for the given MDC assignment.

To compute minimum MDC and SRV-based register assignment, we apply the above subroutine iteratively over all the possible subsets $\mathcal{C}$ of the set of FUs $\mathcal{F}$.

MDC and register assignment :

The maximum number of registers K is given as an input.
Let $S = \emptyset$: a set of pairs of MDC and register assignments;
for (each subset $\mathcal{C}$ of $\mathcal{F}$) {
  $\xi \leftarrow$ SRV-based register assignment for a given $\mathcal{C}$;
  if (the number of registers in $\xi \leq K$) **then** $S \leftarrow (\xi, \mathcal{C})$; }
Output $(\xi, \mathcal{C}) \in S$ having the min $|\mathcal{C}|$ among elements in S;

V. Case Study

We demonstrated design examples on MDC and SRV-based register assignment. As an input application, we used the fifth-order wave digital elliptic filter [11]. To treat the DFG as a DAG, we cut open delay elements, and replaced them with external inputs and outputs. The schedule shown in Fig. 4 is designed with assuming that three adders ADD_1, ADD_2, ADD_3 and one multiplier MUL are available, and every operation is a single-cycle operation. FU assignment is given as, $\rho(O_i (i = 1,2,4,5,7,9,13,16,20,23,29,30)) = \mathrm{ADD}_1$, $\rho(O_i (i = 3,10,12,14,17,21,25,26,31,33)) = \mathrm{ADD}_2$, $\rho(O_i (i = 18,22,27,34)) = \mathrm{ADD}_3$, and all the multiplications are assigned to one MUL.

Fig. 4. Schedule and register assignment for a complete SRV-based design. Symbol "×" represents a control step to which any data cannot be assigned.

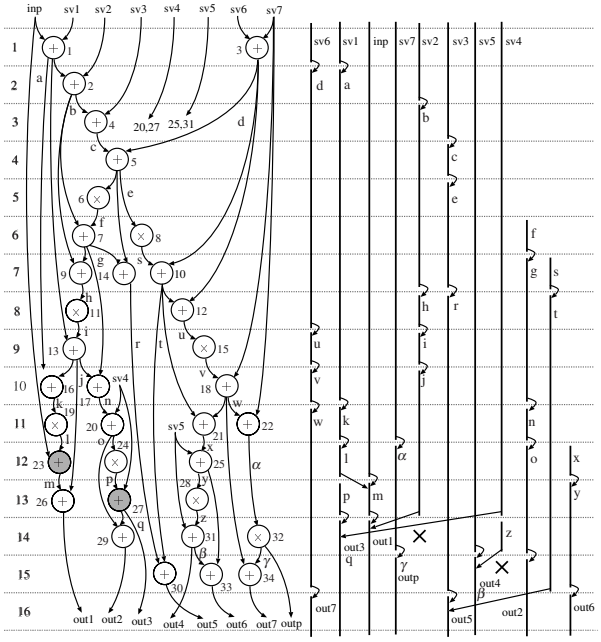

Fig. 5. Schedule and register assignment which are obtained by setting the number of registers to 11.

A. Complete SRV-based Design

We first demonstrate a design example on SRV-based register assignment without MDC technique. The solution is obtained by SRV-based register assignment for a given MDC assignment algorithm in the previous section with $\mathcal{C} = \emptyset$. As we can see from Lemma 1 and the complexity of the algorithm, SRV-based minimum register assignment without MDC technique is in the class **P**.

The time chart shown in Fig. 4 consists of two parts. The left half part (schedule table) shows operation schedule together with dependency arcs. 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 lifetime ℓ_i to ℓ_j means that ℓ_j is the output of the operation which uses ℓ_i lastly. For this problem instance, 12 registers are needed to complete SRV-based register assignment, while 11 registers are enough for a conventional register assignment.

B. MDC and SRV-Based Register Assignment

Fig. 5 shows the result of MDC and SRV-based register assignment, which is obtained by setting the number of registers to 11. Gray operations in the schedule table are operations which require MDC for ensuring the hold constraint, while the hold constraints for the other operations are ensured by SRV-based register assignment. In this case, two adders among four functional units require MDC.

VI. Conclusion

This paper introduced a novel class of datapaths that has robustness against delay variations, which is achieved by MDC and SRV-based register assignment. We proved that the problem to minimize the number of MDCs is **NP**-hard in general, while the problem with a fixed number of FUs is in the class **P**. In this paper, a polynomial time computation algorithm for the problem with a fixed number of FUs was presented. Since the algorithm works in an exponential time in the number of FUs, its application is limited to problem instances with a small number of FUs. Development of an efficient heuristic algorithm for MDC problem for a large number of FUs is one of our future problems.

Acknowledgments 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] 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. CICC*, pp. 582–585, May 2001.

[2] M. Murakawa, E. Takahashi, T. Susa, and T. Higuchi, "Post-fabrication clock timing adjustment for digital LSIs with generic algorithms ensuring timing margins," *Rep. MIRAI Project*, 2004.

[3] J. Jung and T. Kim, "Timing variation-aware high-level synthesis," *Proc. ICCAD 2007*, pp. 424–428, Nov. 2007.

[4] S. Ghosh, S. Bhunia, and K. Roy, "CRISTA: a new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, Issue 11, pp. 1947–1956, Nov. 2007.

[5] J. Xi and W W.-M. Dai, "Jitter-tolerant clock routing in two-phase synchronous systems," *Proc. ICCAD 1996*, pp. 316–320, Nov. 1996.

[6] K. Inoue, M. Kaneko, and T. Iwagaki, "Structural robustness of datapaths against delay-variation," *Proc. SASIMI 2007*, pp. 272–279, Oct. 2007.

[7] K. Inoue, M. Kaneko, and T. Iwagaki, "Novel register sharing in datapath for structural robustness against delay variation," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E91-A, no. 4, pp. 1044–1053, Apr. 2008.

[8] M. Garey and D. Johnson, *Computers and Intractability – A Guide to the Theory of NP-completeness*, W. H. Freeman Publishers, 1979.

[9] K. Inoue, M. Kaneko, and T. Iwagaki, "Complexities and algorithms of minimum-delay compensation problems in datapath synthesis," *Tech. rep. IEICE*, VLD2007-93, pp. 25–30, Nov. 2007 (in Japanese).

[10] F. J. Kurdahi and A. C. Parker, "REAL: a program for register allocation," *Proc. DAC 1987*, pp. 210–215, Jun. 1987.

[11] P. Michel, U. Lauther, and P. Duzy, *The Synthesis Approach to Digital System Design*, Kluwer Academic Publishers, 1992.