社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－137（26）
2008 / 11 / 19

タイミング制御による性能を考慮した耐遅延変動データパス合成

井上 恵介† 金子 峰雄† 岩垣 剛†

† 北陸先端科学技術大学院大学 情報科学研究科 〒923-1292 石川県能美市旭台 1-1
E-mail: †{k-inoue,mkaneko,iwagaki}@jaist.ac.jp

あらまし 回路のホールド・タイミング条件を保証する手法としてレジスタへの書き込み制御信号の到着に相対的な時間順序関係を規定する Backward-Data-Direction (BDD) クロッキングが知られている．本稿ではこれに加えてセットアップ・タイミング条件のためのレジスタへの書き込み制御信号到着順序 Forward-Data-Direction (FDD) クロッキングの必要性をも考慮し，適切なレジスタへの書き込み制御信号到着順序 (順序クロッキング; ordered clocking) を有するデータパス回路を合成する問題について議論する．はじめに遅延変動の下で正しく動作するデータパス合成を，回路が正当な順序クロッキングを有するためのレジスタ割り当て問題として定式化し，レジスタ数を最小化する問題が NP 困難であることを示す．次に，この問題の一解法として整数計画法に基づく解法を提案し，計算機実験によってその有効性を評価する．

キーワード データパス合成，遅延ばらつき，順序クロッキング，レジスタ割り当て，NP 困難，整数計画法

Delay Variability-Aware Datapath Synthesis Based on Safe Clocking for Setup and Hold Timing Constraints

Keisuke INOUE†, Mineo KANEKO†, and Tsuyoshi IWAGAKI†

† School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
E-mail: †{k-inoue,mkaneko,iwagaki}@jaist.ac.jp

Abstract It is known that Backward-Data-Direction (BDD) clocking is one of methods to guarantee the hold timing constraint. BDD clocking is a relative order relations between the arrivals of control signals at a pair of registers. In this paper, in addition to BDD clocking, we introduce Forward-Data-Direction (FDD) clocking for the setup timing constraint, and discuss the synthesis of datapath which has appreciate order relations of the arrivals of control signals between registers (ordered clocking). First, we formulate the problem as a minimum register assignment problem for datapaths which has a proper ordered clocking, and then prove that the problem is NP-hard. After that, we propose an ILP formulation and show the experimental results for some benchmark circuits.

Key words Datapath synthesis, delay variation, ordered clocking, register assignment, NP-hard, ILP

1. 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 [1] [2].

Recently, Backward-Data-Direction (BDD) clocking based design in high-level has been proposed [3] [4]. (In other literatures, it is called "contra"-data-direction clocking. However, here we call it "backward"-data-direction.) BDD clocking is a well-known technique to ensure the hold constraint [5]. In this technique, the arrival of the clock signal at a source register is controlled to be no earlier than that of the clock signal at the destination register. Hence, if only the timing order relation between registers is maintained (for example, by an appropriate choice of a clock tree layout), the satisfaction of the hold constraint is guaranteed. However, on the other hand, applying BDD clocking may degrade the timing mar-

gin for the setup constraint. Forward-Data-Direction (FDD) clocking rather than BDD is preferable to the setup constraint. In this paper, we will discuss an overall framework of "ordered clocking" (including FDD and BDD) together with SRV-based register assignment [6], [7] [8], formulate the problem to determine register assignment and clocking order among registers simultaneously, and present an ILP solution.

This paper is organized as follows. Section 2 describes timing variability-aware design. In Section 3, our optimization problem, performance-aware ordered clocking-based register assignment problem is formulated, and NP-hardness of our problem is proven. After that, we derive ILP formulation in Section 4. Experimental results for some benchmark circuits are shown in Section 5. Finally, Section 6 concludes the paper.

2. Delay Variability-Aware Design

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 dependences 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.

2.1 Setup and Hold Constraints

For a register-transfer level datapath circuit, a register-to-register path corresponds to a multiple-input, multiple-output combinational circuit. It means that a register-to-register path consists of several logic paths. "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.

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) FU_A (we will use ρ to represent functional unit assignment, like $\rho(O_b) = \mathrm{FU}_A$), an input data a for O_b is stored in a register r_1, and the result b of O_b is written to a register r_2. In this paper, we

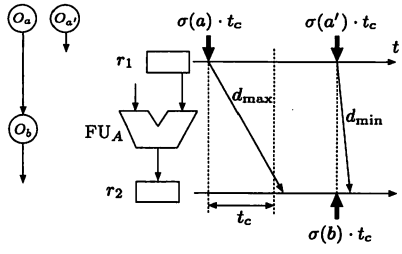


Fig. 1 Setup and hold constraints

assume that a datapath is designed 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 r_i, has the same nominal value $r(i,j) = r_0$ for all i and j. Note that a similar argument can be made for a datapath designed with intentional skew, but we will limit our discussion to a datapath designed with zero skew for the sake of simplicity.

The arrival of the control signal at r_2 for latching data b has to be later than the arrival of b. This is called "setup constraint", and is formulated as

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

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 r_1 to r_2 including the output delay of r_1 and the setup time of r_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 several data. If data a and a' share the same register r_1, and a is overwritten by a', the control signal for latching data b has to arrive at r_2 before b is destroyed by the change of input a to a'. This is called "hold constraint" and is formulated as

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

where $d_{\min}$ is the minimum-path delay from r_1 to r_2 including the output delay of r_1 minus the hold time of r_2.

2.2 Coping with Delay Variation

In this paper, we mainly focus on the variation of the arrival timing of a control signal at a register. The following shows the setup and hold constraints considering delay variations,

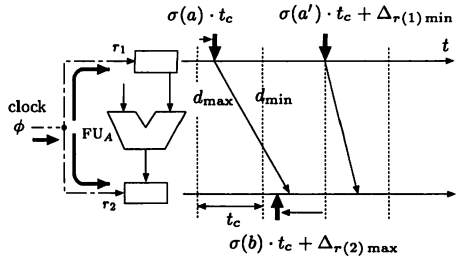
$$\sigma(a) \cdot t_c + \Delta_{r(1)\max} + d_{\max} + \leqq \sigma(b) \cdot t_c + \Delta_{r(2)\min}, \quad (1)$$
$$\sigma(b) \cdot t_c + \Delta_{r(2)\max} < \sigma(a') \cdot t_c + \Delta_{r(1)\min} + d_{\min} \quad (2)$$

where $\Delta_{r(i)\max} = \max_j\{\Delta_{r(i,j)}\}$ and $\Delta_{r(i)\min} = \min_j\{\Delta_{r(i,j)}\}$ with $\Delta_{r(i,j)}$ as the delay variation of $r(i,j)$.

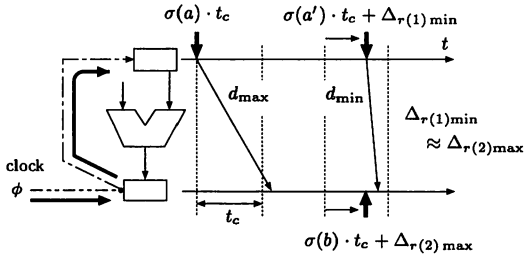
With respect to the setup constraint, a feasible clock period t_c is determined by the details of $d_{\max}$ and $\Delta_{r(i)\max/\min}$. In a conventional design, designers must insert a sufficient timing margin for delay variations, and it causes a performance degradation.

(C1 : Ensuring the hold constraint by SRV)

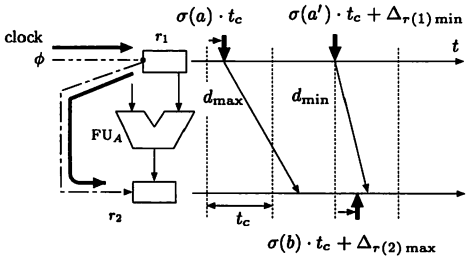
Recently, Structural Robustness against delay Variation (SRV)-based register assignment has been proposed [6] [7] [8]. SRV-based register assignment provides more than or equal to one step margin for the hold constraint of O_b. Fig. 3(a) shows a register assignment based on (C1). In this figure, an oval represents a scheduled operation, a



(a) Nominal design



(b) Backward-Data-Direction clocking



(c) Forward-Data-Direction clocking

Fig. 2 Forward-/Backward- Data Direction clocking.

rectangle represents data-lifetime, a directed edge (u, v) between lifetimes u and v means that v is the output of the operation that uses u lastly, and data in the gray region are data assigned to the same register. When O_b is the sole operation which uses a as an input, the writing an output to one of input registers (Fig. 3(b)) is done safely under delay variation.

(C2 : Ensuring the hold constraint by BDD clocking) "Guaranteeing $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ for delay variations by BDD clocking." From (2), the following inequality holds.

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

If $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is guaranteed, the left hand side of (2)' is always negative. Supposing $0 < d_{\min}$ is trivial, (2)' is satisfied with $\sigma(a') \geqq \sigma(b)$ under any delay variation (Fig. 2(b)). The condition $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is the Backward-Data-Direction (BDD) clocking. Fig. 4 shows the register assignment based on (C2).

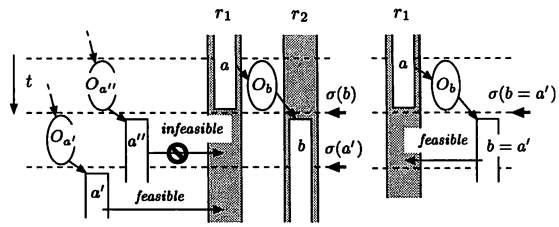


Fig. 3 Illustration of C1: Ensuring the hold constraint by SRV

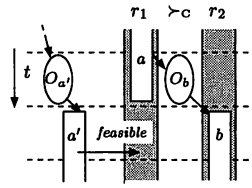


Fig. 4 Illustration of C2: Ensuring the hold constraint by BDD

(C3 : Ensuring the setup constraint by FDD clocking) "Guaranteeing $\Delta_{r(1)\max} < \Delta_{r(2)\min}$ for delay variations by FDD clocking."

If the nominal setup constraint $\sigma(a)\cdot t_c + d_{\max} \leqq \sigma(b)\cdot t_c$ is "tight", it can be easily violated dependence on the values of $\Delta_{r(1)\max}$ and $< \Delta_{r(2)\min}$. However, if $\Delta_{r(1)\max} < \Delta_{r(2)\min}$ is guaranteed, we have only to maintain delay variations of datapath circuits so that $(d_{\max})/(\sigma(b) - \sigma(a)) < t_c$ is satisfied without considering delay variations of control signals (Fig. 2(c)). The condition $\Delta_{r(1)\max} - \Delta_{r(2)\min} < 0$ is the Forward-Data-Direction (FDD) clocking technique for the setup constraint.

Here we formally define "tight operation" as follows. For an operation O_b, if its result b (which is not destroyed by other data until $\sigma(b)\cdot t_c$) arrives at its destination register no earlier than one step before $\sigma(b)$, we call O_b a tight operation. If O_b is a "tight operation", we must apply FDD clocking to input-output register pair for O_b. Fig.s 5(a) and 5(b) show the register assignment based on (C3).

3. Notation and Problem Formulation

In this section, we formulate our problem and show a simple demonstrative example.

3.1 Notation

For simplicity in notation, we use $r_i \succ_C r_j$ for two registers r_i, r_j to represent the timing order $\Delta_{r(i)\min} > \Delta_{r(j)\max}$. Then, ordered clocking (including BDD and FDD) can be represented as a partial order $\succ_C$ on a set of registers $\mathcal{R}$, and we call it clocking-order. It is clear that clocking-order

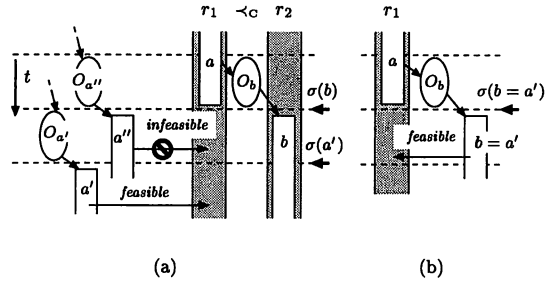


Fig. 5 Illustration of C3: Ensuring the hold constraint by FDD

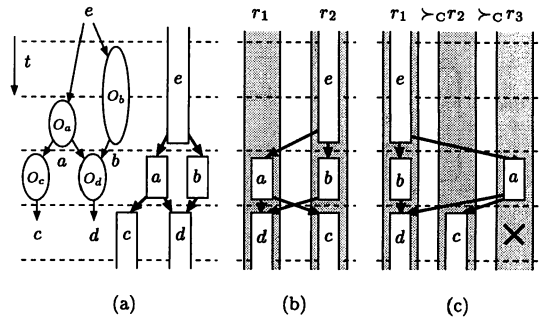


Fig. 6 A problem instance (a), and two different feasible minimum register assignments (b) and (c) for a conventional design, an OC-based design, respectively.

$(\mathcal{R}, \succ_C)$ must be transitive ($r_i \succ_C r_j, r_j \succ_C r_k \Rightarrow r_i \succ_C r_k$) and antisymmetric ($r_i \succ_C r_j \Rightarrow r_j \not\succ_C r_i$), but not be reflective ($\forall r \in \mathcal{R}$, $r \not\succ_C r$) (because of it, clocking-order is not a partial order in a strict sense, but just a quasi-order). The antisymmetric law means that a cycle of order relation such as $r_1 \succ_C r_2, \ldots, r_{k-1} \succ_C r_k, r_k \succ_C r_1$ is not allowed.

3.2 Problem Formulation

Our optimization problem, performance-aware ordered clocking (OC)-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 representing dependences between operations, (2) a set of data $\mathcal{D}$, which has one-to-one correspondence to $\mathcal{O}$, (3) a control schedule $\sigma : \mathcal{O} \to \mathbb{Z}_+$, (4) sets of functional units $\mathcal{F}$ and registers $\mathcal{R}$, and (5) FU assignment $\rho : \mathcal{O} \to \mathcal{F}$ as a problem instance, and outputs a register assignment $\xi : \mathcal{D} \to \mathcal{R}$ together with clocking-order on $\mathcal{R}$ so that setup and hold constraints in the application are ensured by (C1)~(C3), and $|\mathcal{R}|$ is minimized.

Note that whether each operation is a tight operation or not depends on a control schedule, an FU assignment, and an MUX control. Now, our problem receives control schedule and FU assignment. Assuming that an MUX switching is done appropriately, operation type (tight or not) is decided.

3.3 Example

We explain our OC-based register assignment problem using a small DFG and its schedule shown in Fig. 6(a). We prepare three functional units (two ALUs ALU_1, ALU_2 and one multiplier MUL), and assume that O_a, O_c, and O_d are additions (single-cycle operations), O_b is a multiplication (two-cycle operation), and the functional unit assignment is done as $\rho(O_a) = \rho(O_c) = ALU_1$, $\rho(O_d) = ALU_2$, and $\rho(O_b) = \text{MUL}$.

In the first assignment shown in Fig. 6(b), which uses the minimum number of registers, data e in r_2 is overwritten by b at the same timing that output a of O_a is written to r_1. If the arrival of the control signal to write a to r_1 is later than the arrival of the fastest effect of the input change from e to b, incorrect data polluted by this fastest effect may possibly be written to r_1. To avoid this malfunction, we have to keep $r_2 \succ_C r_1$, i.e., $\Delta_{r(2)\min} > \Delta_{r(1)\max}$, for the hold guarantee of O_a, and at the same time, $r_1 \succ_C r_2$, i.e., $\Delta_{r(1)\min} > \Delta_{r(2)\max}$, for the hold guarantee of O_c. Consequently, we have $r_1 \succ_C r_1$ and $r_2 \succ_C r_2$ by the transitive law in $(\mathcal{R}, \succ_C)$, but it is infeasible because order relation $r_i \succ_C r_i$ is forbidden by non-reflective law.

Next, we show that the second assignment shown in Fig. 6(c) is a minimum register assignment for OC-based design. Any data whose start time is same with the end of a cannot share the same register with a because O_c and O_d are setup-tight operations which use a lastly (symbol "×" in Fig. 6(c) means a forbidden step to which any data cannot be assigned). So assuming that c, d, and a are written to r_1, r_2, and r_3, respectively, we need clocking-order $r_2 \succ_C r_3$ and $r_1 \succ_C r_3$ by FDD clocking rule. These clocking-orders can be written as $r_1 \succ_C r_2 \succ_C r_3$ from the transitive law, and the hold constraint of O_a can be ensured by this clocking-order. Note that O_b is setup-tight and the operation which uses e lastly, however, the setup and hold constraints of b are ensured by written-back to the input register, i.e. sharing r_1 with e, and the hold constraints of c and d are ensured by one-step margin (SRV property). Therefore these clocking-order assignments are feasible, and every setup and hold constraint is ensured by ordered clocking and SRV. The number of registers needed for OC-based design is determined by the maximum number of temporal-overlaps of lifetimes and symbol "×"s. So we can conclude that the assignment is minimum.

This example shows that a conventional register assignment is not always a feasible one for an OC-based register assignment, and it tends to increase the number of registers.

3.4 Computational Complexity

In [4], the safe clocking minimum register assignment problem (only consider BDD clocking) is formulated and shown

to be NP-hard. Our problem, performance-aware OC-based register assignment problem includes it as sub-problem. Therefore we have the following theorem.

Theorem 1: Performance-aware OC-based register assignment problem is NP-hard.

4. ILP Formulation

At the first step, let n_R be the maximum number of registers available, $\mathcal{R}$ be a set of registers $\{r_1, r_2, \ldots, r_{n_R}\}$, and $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ be a set of data needed to be assigned to registers. Without loss of generality, we can assume the OC-order relation on $\mathcal{R}$ as

$$r_1 \succ_C r_2 \succ_C \cdots \succ_C r_{n_R}.$$

(1) For each i ($1 \leqq i \leqq n$) and j ($1 \leqq j \leqq n_R$), we prepare a binary variable $x_{i,j} \in \{0,1\}$. $x_{i,j} = 1$ if data d_i is assigned to register r_j, otherwise $x_{i,j} = 0$. Each data d_i must be assigned to exactly one register. This condition can be represented by the equation

$$\sum_{j=1}^{n_R} x_{i,j} = 1, \quad 1 \leqq i \leqq n.$$

(2) If the lifetimes of data d_i and d_k have an overlap, they cannot share a register. This condition can be represented by the inequality

$$x_{i,j} + x_{k,j} \leqq 1, \quad 1 \leqq j \leqq n_R.$$

(3) Let d_k be an output data of an operation that uses data d_i lastly, and there is another data d_ℓ whose start time of its lifetime is the same with d_k ($\ell \neq k$). From the register sharing conditions for OC-based register assignment and the specified OC-order on $\mathcal{R}$, if the index of the register to which d_k is assigned is larger than that for d_i (i.e., $\xi(d_i) \succ_C \xi(d_k)$), d_i and d_ℓ can share the same register. On the other hand, if the index of the register to which d_k is assigned is smaller than that for d_i (i.e., $\xi(d_k) \succ_C \xi(d_i)$), d_i and d_ℓ cannot share the same register This condition can be represented by the inequality

$$x_{i,j} + x_{\ell,j} \leqq 1 + \sum_{m=j+1}^{n_R} x_{k,m}, \quad 1 \leqq j \leqq n_R - 1.$$

(4) Let d_k be an output data of a setup-tight operation that uses data d_i. The index of the register to which d_k is assigned is smaller than that for d_i (i.e., $\xi(d_k) \succ_C \xi(d_i)$). This condition can be represented by the inequality

$$\sum_{j=1}^{n_R} j \cdot x_{i,j} < \sum_{j=1}^{n_R} j \cdot x_{k,j}.$$

(5) For representing our objective, we prepare a binary variable $z_j \in \{0,1\}$ for each j, $1 \leqq j \leqq n_R$. $z_j = 1$ if at least one data is assigned to register r_j, otherwise $z_j = 0$. This condition can be represented by the inequality

$$x_{i,j} \leqq z_j, \ 1 \leqq i \leqq n, 1 \leqq j \leqq n_R.$$

(6) Our objective is to minimize the number of registers needed in OC-based register assignment, which can be written to

$$\text{minimize} \ \sum_{j=1}^{n_R} z_j.$$

5. Experimental Results

In this section, we demonstrate some design examples. DFGs used in the experiments are the following three benchmark circuits: the 8-point Fast Fourier Transform (8-FFT) [10], the fifth-order Elliptic Wave digital Filter (EWF) [11], and the Fast Discrete Cosine Transform (FDCT) [12]. In Table 1, the column "design" represents benchmark circuits. We use two types of functional units, ALU (addition/subtraction) and MUL (multiplication). ALU (MUL) is treated as a single- (two-) cycle operation, respectively. For each benchmark circuit, we picked up three palate optimal points in the numbers of ALUs, MULs, and latency (defined as total control steps). The column "FUs" represents a set of functional units for each palate point. We use list scheduling heuristic algorithm to obtain schedule results, and the column "latency" represents latency.

5.1 Conventional Design

We first demonstrate a design example on a conventional register assignment. In this case, minimum register assignment problem is known to be solved in a polynomial time by the well-known left-edge algorithm [9]. The column "conventional design / # reg." represents the number of registers obtained by the algorithm.

For each setup and hold constraint of operation, we count the number of the setup and hold constraints of operations which do not satisfy the design rules described in Section 2.2 (hence, they can be potentially violated). The column "conventional design / potential violations /setup (hold)" represents the number of potential violations of setup (hold, respectively) constraints. The column "conventional design / potential violations /total" represents the total number of setup and hold potential violations. Note that we consider the setup and hold constraints for each pair of input and output data of an operation. For example, if operation O_a receives data b (stored in register r_1) and c (stored in register r_2) and outputs data a (stored in register r_3), we consider the setup and hold constraints between r_1 and r_3, and between r_2 and r_3. The result shows that 30 to 70 % of the setup

Table 1 Experimental results

design (#op)	FUs*	latency	conventional design				OC-based design				
			#reg	potential violations‡			#reg	potential violations‡			ILP
				setup	hold	total		setup	hold	total	time[s]
8-FFT (29)	(2,1)	12	10	40/58 (69%)	19/40 (23%)	59/98 (60%)	15	0/58 (0%)	0/40 (0%)	0/98 (0%)	1501.83
	(3,2)	9	11	41/58 (70%)	20/40 (50%)	61/98 (62%)	15	0/58 (0%)	0/40 (0%)	0/98 (0%)	6909.25
	(4,2)	8	10	41/58 (71%)	31/40 (78%)	72/98 (73%)	15/14†	0/58 (0%)	0/40 (0%)	0/98 (0%)	10000<
EWF (34)	(2,1)	21	10	37/68 (54%)	10/53 (19%)	47/121 (39%)	17	0/68 (0%)	0/53 (0%)	0/121 (0%)	263.12
	(3,2)	18	10	33/68 (49%)	9/52 (17%)	42/120 (35%)	18	0/68 (0%)	0/52 (0%)	0/120 (0%)	335.88
	(3,3)	17	11	32/68 (47%)	9/51 (18%)	41/119 (34%)	19/18†	0/68 (0%)	0/51 (0%)	0/119 (0%)	10000<
FDCT (42)	(2,2)	20	12	45/84 (54%)	23/65 (35%)	68/149 (46%)	17/16†	0/84 (0%)	0/65 (0%)	0/149 (0%)	10000<
	(3,1)	34	11	41/84 (49%)	13/66 (20%)	54/150 (36%)	14	0/84 (0%)	0/66 (0%)	0/150 (0%)	84.10
	(3,3)	14	11	49/84 (58%)	27/64 (42%)	76/148 (51%)	17/16†	0/84 (0%)	0/64 (0%)	0/148 (0%)	10000<

*(#ALU,#MUL), ‡#violated constraints/#constraints (ratio %), † temporary best/lower-bound

and hold constraints are potential violations. It can be easily expected that the operation type (tight or not) strongly depends on the schedule and FU assignment. Even in a conventional design, potential violations may be decreased in part by modifying schedule and FU assignment.

5.2 OC-Based Design

We next demonstrate a design example on OC-based design. For each problem instance, ILP constraints are generated by a computer program written in C language, gcc 2.8.1 on Sun Blade 1500. ILP is solved by CPLEX 11.0.0 [13] on a PC equipped with 2.40 GHz AMD (R) Dual Opteron (TM), 8.00 GB RAM. The column "OC-based design / # reg" represents the number of registers obtained by ILP, and the column "OC-based design / time[s]" represents the total execution time of ILP. In some cases, the solver cannot terminate over 10000 seconds (denoted as "10000<" in the table), and we show a temporary best solution and lower bound in this case. It motivates us to develop an effective heuristic algorithm. In OC-based design, the number of registers needed to each benchmark increases by about 50 % compared with a conventional design. Instead of the drawback, we can obtain a datapath which has no potential violations of the setup and hold constraints. Extra registers are introduced partly in order to compensate tight operations, and they may be decreased by modifying schedule and FU assignment.

6. Conclusion

This paper introduced a novel class of datapaths that have robustness against delay variations, which is achieved by ordered clocking (OC)-based register assignment. We have proven that the problem to minimize the number of registers is NP-hard. In this paper, an integer linear programming formulation for this novel NP-hard problem was presented. Experimental results showed the effectiveness of our approach. Development of an efficient heuristic algorithm for our problem for a large problem instance 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] J. Jung and T. Kim, "Timing variation-aware high-level synthesis," *Proc. International Conference on Computer-Aided Design (ICCAD)*, pp. 424–428, November 2007.

[2] 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, November 2007.

[3] K. Inoue, M. Kaneko, and T. Iwagaki, "Safe clocking minimum register assignment in high-level synthesis," *IEICE Technical Report*, vol. 108, no. 105, pp. 7–12, June 2008 (in Japanese).

[4] K. Inoue, M. Kaneko, and T. Iwagaki, "Safe clocking register assignment in datapath synthesis," *Proc. International Conference on Computer Design (ICCD)*, pp. 120–127, October 2008.

[5] N H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design - System Perspective, Second Edition*, Addison-Wesley Publishing Company, 1994.

[6] K. Inoue, M. Kaneko, and T. Iwagaki, "A basic study on datapath synthesis considering delay variation," vol. 106, no. 387, November 2006 (in Japanese).

[7] K. Inoue, M. Kaneko, and T. Iwagaki, "Structural robustness of datapaths against delay-variation," *Proc. Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, pp. 272–279, October 2007.

[8] 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, April 2008.

[9] F. J. Kurdahi and A. C. Parker, "REAL: a program for register allocation," *Proc. Design Automation Conference (DAC)*, pp. 210–215, June 1987.

[10] B. Mulgrew, P. Grant, and J. Thompson, *Digital Signal Processing*, Macmillan Press Ltd, 1999.

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

[12] I. Ahmad, M. K. Dhodhi, and F. M. Ali, "TLS: a tabu search based scheduling algorithm for behavioral synthesis of functional pipelines," *BCS The Computer Journal*, vol. 43, no. 2, pp. 152–166, 2000.

[13] ILOG, CPLEX, http://www.ilog.com