Safe Clocking Register Assignment in Datapath Synthesis

Keisuke Inoue, Mineo Kaneko, and Tsuyoshi Iwagaki
Japan Advanced Institute of Science and Technology
{*k-inoue, mkaneko, iwagaki*}*@jaist.ac.jp*

***Abstract*— For recent and future nanometer-technology VLSIs, static and dynamic delay variations become a serious problem. In many cases, the hold constraint, as well as the setup constraint, becomes critical for latching a correct signal under delay variations. While the timing violation due to the fail of the setup constraint can be fixed by tuning a clock frequency or using a delayed latch, the timing violation due to the fail of the hold constraint cannot be fixed by those methods in general. Our approach to delay variations (in particular, the hold constraint) proposed in this paper is a novel register assignment strategy in high-level synthesis, which guarantees safe clocking by contra-data-direction (CDD) clocking. After the formulation of this new register assignment problem, we prove NP-hardness of the problem, and then derive an integer linear programming formulation for the problem. The proposed method receives a scheduled data flow graph, and generates a datapath having (1) robustness against delay variations, which is ensured by CDD-based register assignment, and (2) the minimum possible number of registers. Experimental results show the effectiveness of the proposed method for some benchmark circuits.**

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). [3] proposed a performance yield constrained high level synthesis based on a statistical static timing analysis (SSTA). [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 [5]. Furthermore, while the timing violation due to the fail of the setup constraint can be fixed by tuning a clock frequency or using a delayed latch [4], the timing violation due to the fail of the hold constraint cannot be fixed by those methods in general.

Typical circuitry to ensure the hold constraint is the "double-latch" which uses two non-overlap clocks, one drives master latches and the other drives slave latches [6]. 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.

Contra-data-direction (CDD) clocking [7] and structural robustness against delay variation (SRV)-based register assignment [8] [9] are other techniques to ensure the hold constraint. Our motivational question is whether there exists a pair of the assignment of data to registers (which determines data-direction between registers) and the assignment of timing order to registers which achieves the CDD clocking. Starting with this question, first we formulate the CDD-based register assignment problem along the scenario of delay variability-aware datapath design. After that, we discuss the problem complexity and solution algorithm. Despite a simple feature of the original CDD clocking, the CDD-based minimum register assignment problem is proven to be NP-hard. As a first attempt to approach this novel NP-hard problem, we derive an integer linear programming (ILP) formulation of this CDD-based minimum register assignment problem. A resultant datapath has (1) robustness against delay variations, which is ensured by CDD-based register assignment, and (2) the minimum possible number of registers. Of course, in a practical datapath design, several metrics in addition to the number of registers are chosen as design objectives. Our ILP formulation of the CDD-based register assignment can be easily built into other ILP-based datapath optimization.

This paper is organized as follows. In Section II, we describe related works. Section III describes timing variability-aware design. In Section IV, CDD-based register assignment problem is formulated, and NP-hardness of our problem is proven. After that, we derive ILP formulation in Section V. Experimental results for some benchmark circuits are shown in Section VI. Finally, Section VII concludes the paper.

II. Related Works

Contra-data-direction (CDD) clocking is a well-known technique to ensure the hold constraint [7]. 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 (Fig. 1). Hence, if only the timing order relation between registers is maintained (for example, by an appropriate choice of a clock tree layout), the satisfaction

978-1-4244-2658-4/08/$25.00 ©2008 IEEE

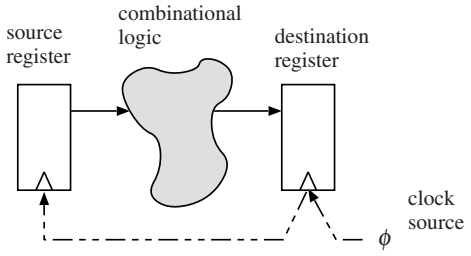

Fig. 1. Contra-data-direction (CDD) clocking. The arrival of the clock signal at a source register is controlled to be no earlier than that of the clock signal at the destination register.

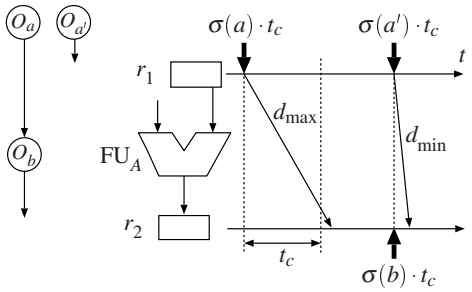

Fig. 2. Setup and hold constraints

of the hold constraint is guaranteed. However, if the data flow between two registers is bi-directional, the timing order assignment comes to a deadlock.

Recently, another approach to ensure the hold constraint under delay variation has been proposed at a higher level [8] [9]. This approach is based on a kind of constrained register assignment, which is named structural robustness against delay variation (SRV)-based register assignment, and it makes more than one clock margin for the hold constraint. However this technique tends to need more registers than a conventional register assignment. In this paper, we will discuss the CDD clocking in datapath synthesis for designing a cost effective datapath having robustness against delay variations, where timing order deadlocks are avoided by incorporating SRV-based register assignment.

III. Timing 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.

A. 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. 2 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 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 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

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

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.

B. Coping with Delay Variation

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,

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

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

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 $r(i,j)$.

With respect to the setup constraint, the details of $d_{\max}$, $\Delta_{d_{\max}}$, and $\Delta_{r(i)\max/\min}$ rely partly on the designers' choices (i.e., designers' efforts) done in the downstream logic design and layout design, and $\sigma(a) < \sigma(b)$, which is a mandatory choice in datapath high-level design, guarantees the existence of a proper t_c which satisfies the setup constraint under any delay values.

However, it is not the case for the hold constraint. In zero skew design (Fig. 2), the violation of hold constraint for writing b to register r_2 is due to the increase of $-\Delta_{r(1)\min} + \Delta_{r(2)\max}$ in (2) (Fig. 3(a)). To ensure the hold constraint, the followings are design constraints used in this paper for coping with delay variation.

(C1 : ensuring the hold constraint by SRV)
"Disallowing to assign data a' to r_1 at the same timing with $\sigma(b)$." It corresponds to more than equal to one step margin for the hold constraint of O_b (Fig. 3(b)). Note that several steps margin might be needed for actual variations. However, even in that case, we can do the same discussion. Therefore, we assume that one step margin is enough for the hold constraint for simplicity. Fig. 4 shows the register assignment based on (C1). In this figure, an oval represents a scheduled operation, a 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. Such a register assignment (including the next (C2)) is called structural robustness against delay variation (SRV)-based register assignment [8][9].

(C2 : ensuring the hold constraint by SRV) "Writing back b to r_1 $(r_2 = r_1)$." When O_b is the sole operation which uses a as an input, $a' = b$ holds. As a result, by substituting $\sigma(a') = \sigma(b)$ into the hold constraint, (2) is reduced to

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

where $\Delta_{r(2=1)\max} - \Delta_{r(1)\min}$ represents the maximum timing difference (with respect to the same clock edge) between FFs in one register. Assuming $\Delta_{r(2=1)\max} - \Delta_{r(1)\min}$ is negligible small compared with $d_{\min} + \Delta_{d_{\min}}$, the writing an output to one of input registers (Fig. 5(a)) is done safely under delay variation. On the other hand, when there are several operations which uses a as input lastly, a and the outputs of these operations cannot share the same register (Fig. 5(b)).

(C3 : ensuring the hold constraint by CDD clocking)
"Guaranteeing $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ for delay variations." From (2), the following inequality holds.

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

If $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is guaranteed, the left hand side of $(2)'$ is always negative. Supposing $0 < d_{\min} + \Delta_{d_{\min}}$ is trivial, $(2)'$ is satisfied with $\sigma(a') \geq \sigma(b)$ under any delay variation (Fig. 3(c)). The condition $\Delta_{r(1)\min} > \Delta_{r(2)\max}$ is no less than the contra-data-direction (CDD) clocking. 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, CDD clocking can be represented as a partial order $\succ_c$ on a set of registers $\mathcal{R}$, and we call it CDD-order. It is clear that CDD-order $(\mathcal{R}, \succ_c)$ 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, CDD-order is not a partial order in a strict sense, but just a quasi-order). Note that the antisymmetric law means that a cycle of order relation such as $r_1 \succ_c r_2, r_2 \succ_c r_3, \ldots, r_{k-1} \succ_c r_k, r_k \succ_c r_1$ is not allowed. Fig.s 6(a) and 6(b) show the register assignment based on (C3).

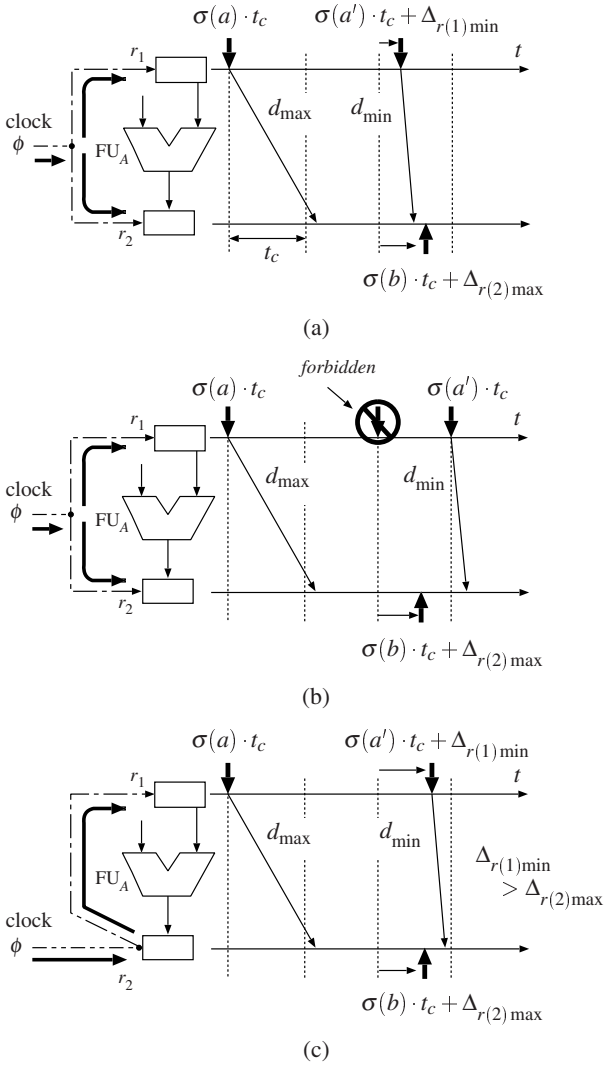

Fig. 3. The hold constraint in (a) is violated by the variation of the arrival of control signals, on the other hand, the hold constraint in (b) is ensured by SRV-based register assignment. In (c), the hold constraint is ensured by CDD clocking.

IV. CDD-Based Register Assignment Problem

In SRV-based register assignment, it tends to increase the number of registers. On the other hand, we carefully define CDD-order with keeping antisymmetric law. In this section, we formulate our problem and show a simple demonstrative example.

A. Formulation

Our optimization problem, CDD-based register assignment problem receives (1) an application algorithm presented by

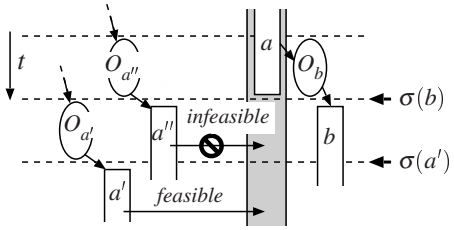

Fig. 4. Feasible register sharing of a and a', a and a'' (C1).

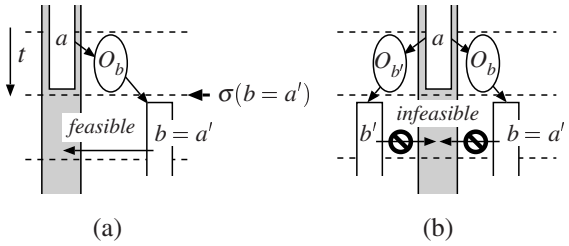

Fig. 5. Feasible register sharing of a and b, a and b' (C2).

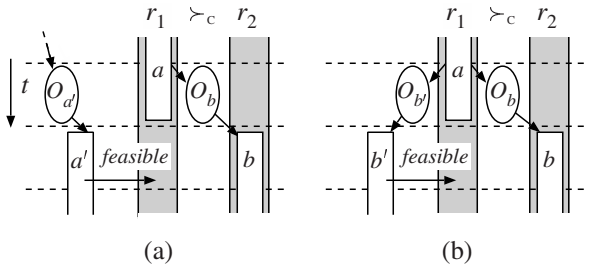

Fig. 6. Feasible register sharing of a and a' (C3).

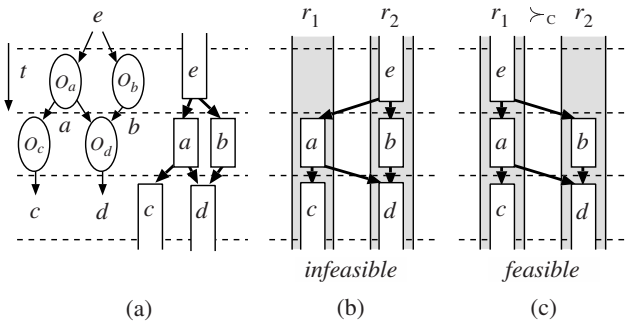

Fig. 7. A problem instance (a), and two different feasible minimum register assignments (b) and (c) for a conventional design. For realizing safe clocking, only assignment (c) with CDD-order $r_1 \succ_c r_2$ is feasible for execution of O_b and O_d.

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 CDD-order on $\mathcal{R}$ so that every hold constraint in the application is ensured by SRV-based register assignment (C1) and (C2), or by CDD clocking (C3), and the number of registers $|\mathcal{R}|$ is minimized.

B. Feasible Register Assignments

In this section, a small demonstrative example of our problem is exhibited using a scheduled DFG shown in Fig. 7(a). Fig.s 7(b) and 7(c) are two minimum feasible register assignments for a conventional register assignment. However, only the assignment in Fig. 7(c) is the feasible one for our problem.

In the first assignment shown in Fig. 7(b), 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 b 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_d. 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. On the other hand, for the second assignment shown in Fig. 7(c), we have to keep $r_1 \succ_c r_2$ for the hold guarantee of O_d, and this CDD-order also ensures the hold constraint for O_b.

This example shows that the conventional register assignment is not always a feasible one for a CDD-based register assignment.

C. Computational Complexity

In a conventional design, if an input DFG is restricted to a directed acyclic graph (DAG), register assignment problem can be solved in polynomial time by using well-known left-edge algorithm [10]. On the other hand, for a given DFG with loop-back dependences, this problem is known to be NP-hard.

With respect to our CDD-based register assignment problem, we have the following theorem.

Theorem 1: CDD-based register assignment problem is NP-hard even if an input DFG is restricted to a DAG.

In order to prove Theorem 1, we have derived a polynomial time reduction from the "modified circular arc coloring problem," which is defined from the circular arc coloring problem [11], to the decision version of our problem. The proof of Theorem 1 is shown in the appendix.

V. ILP Formulation

As a first attempt to approach our novel NP-hard problem, we derive an integer linear programming (ILP) formulation of this CDD-based minimum register assignment problem. In a practical datapath design, several metrics in addition to the number of registers are chosen as design objectives. Our ILP formulation of the CDD-based register assignment can be easily built into other ILP-based datapath optimization.

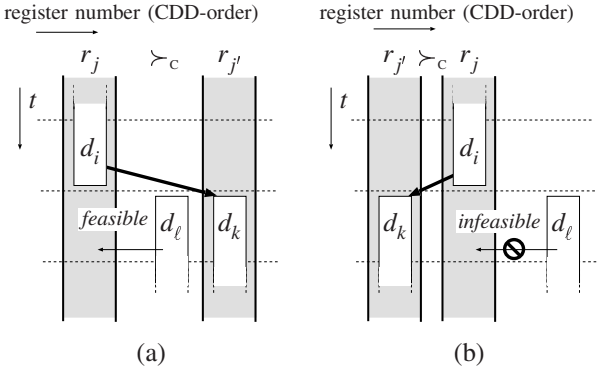

Fig. 8. Feasible register sharing for data d_i and d_ℓ.

At the first step, let n_R be the minimum number of registers needed when all the hold constraints of operations are ensured by SRV-based register assignment. In the case that an input DFG is a DAG, a polynomial time algorithm to compute n_R has been proposed [9]. On the other hand, if it is not the case, we can derive an ILP formulation for SRV-based register assignment problem. For a lack of space, details of this formulation are omitted here. In any way, we can have n_R, and use it as an apparent upper bound of the number of registers needed to CDD-based register assignment. Let $\mathcal{R}$ 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 CDD-order relation on $\mathcal{R}$ as

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

(1) For each i $(1 \le i \le n)$ and j $(1 \le j \le n_R)$, 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

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

(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

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

(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 $(\ell \ne k)$. From the register sharing conditions for CDD-based register assignment and the specified CDD-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 by (C3) in Section III-B (Fig. 8(a)). 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 (Fig. 8(b)). This condition can be represented by the inequality

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

(4) For representing our objective, we introduce a virtual data $e_{\max}$ and a binary variable $w_{\max,j}$ for $e_{\max}$, and we prepare one more register r_{n_R+1} besides $\mathcal{R}$. $w_{\max,j}$ is such a variable that $w_{\max,j} = 1$ if $e_{\max}$ is assigned to register r_j, otherwise $w_{\max,j} = 0$. We assume that $e_{\max}$ is assigned to the register having the largest index among all the data in $\mathcal{D}$. This condition can be represented by the inequality

$$\sum_{j=1}^{n_R} j \cdot x_{i,j} < \sum_{j=1}^{n_R+1} j \cdot w_{\max,j}, \quad 1 \le i \le n.$$

Our objective is to minimize the number of registers needed in CDD-based register assignment. In other words, it is to minimize the index of the register to which $d_{\max}$ is assigned, which can be written to

$$\text{minimize} \sum_{j=1}^{n_R+1} (j-1) \cdot w_{\max,j}.$$

We briefly summarize the complexity of our ILP formulation. First, the number of binary variables $x_{i,j}$ is $n \cdot n_R$. The number of constraints in (2) , (3) , and (4) is $O(n^2 \cdot n_R)$, $O(n^2 \cdot n_R)$, and $n \cdot n_R$, respectively. Therefore, the total number of constraints can be estimated as $O(n^2 \cdot n_R)$.

VI. Experimental Results

In this section, we demonstrate several design examples. The DFGs used in the experiments are the following five benchmark circuits: the 8-point Fast Fourier Transform (8-FFT) [12], the 16-point Fast Fourier Transform (16-FFT) [12], the fifth-order elliptic wave digital filter (EWF) [13], and the inverse discrete cosine transform with column-wise decomposition (IDCT-c) [14]. The original EWF has loop-back dependences. To treat it as a DAG, we cut open delay elements, and replaced them with external inputs and outputs. To distinguish it from the original EWF, we call it EWF DAG-version (EWF-d). We use three types of functional units, ALU (addition/subtraction), MUL (multiplication), and SHT (shift). ALU and SHT are treated as single-cycle functional units, and MUL is treated as a two-cycle functional unit. For each benchmark circuit, we picked up three palate optimal points in the numbers of ALUs, MULs, and SHTs, and latency (defined as total control steps). We use list scheduling heuristic algorithm to obtain schedule results. 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 [15] on a PC equipped with 3.00 GHz Intel(R) Core (TM) 2 Duo processor, 2.00 GB RAM.

Table I shows the experimental results. The column "conv. (#reg)" represents the minimum number of registers for a conventional register assignment, and the column "SRV

(#reg)" represents the number of registers obtained by SRV-based register assignment algorithm proposed in [9]. This algorithm cannot apply to a DFG with loop-back dependences (EWF is the case in our experiment). In all other cases, the algorithm terminated within a few seconds.

On the other hand, the results of CDD-based register assignment is shown in the right half part of Table I. The column "#reg" represents the number of registers obtained by ILP, the column "pre." represents a preprocessing time including ILP constraints generating time and CPLEX reading time, the column "run" represents CPLEX computation time, and the column "total" represents summation of "pre" and "run". Almost all the cases, we have the optimal solution in a reasonable computation time. However for the case 16-FFT with resource set (16,4,0), CPLEX did not finish within 60 hours (denoted as "N/A" in the table). The larger the number of operations which execute concurrently, the larger the number of ILP constraints for CDD clocking (case (3) in Section V). We conjecture that the number of ILP constraints for CDD clocking for this instance is too large for CPLEX to terminate legally. The temporary best solution up till then is 21, which is the lower bound plus 5. Note that, in the case 8-FFT with resource set (2,1,0), SRV-based register assignment is done with the minimum possible number of registers. Therefore we do not have to apply CDD-based register assignment.

Fig. 9 shows an input scheduled DFG and the result of CDD-based register assignment. The left half part (schedule table) shows operation schedule together with dependency arcs. Positive integers represent operations. Every multiplication in the algorithm has a variable multiplicand and a constant multiplier, and we have omitted the constant multiplier from the data flow graph. 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. Integer i corresponds to the output of operation O_i. A directed edge from lifetime i to j means that data j is the output of the operation which uses data i lastly. For this problem instance, register assignment is done with 11 registers which is the minimum possible number of registers.

TABLE I

EXPERIMENTAL RESULTS

design (#op)	resources*	latency (#step)	conv. (#reg)	SRV (#reg)	CDD-based register assignment			
					#reg	time [s]		
						pre.	run	total
8-FFT (29)	(2,1,0)	12	12	12	–	–	–	–
	(3,2,0)	9	11	13	11	0.03	0.12	0.15
	(6,2,0)	7	10	14	10	0.07	0.28	0.35
EWF-d (34)	(2,1,0)	21	10	11	10	0.06	0.06	0.12
	(2,2,0)	19	11	12	11	0.04	0.05	0.09
	(4,4,0)	17	11	13	11	0.04	0.05	0.09
EWF (34)	(2,1,0)	21	10	–	10	0.05	2.04	2.09
	(2,2,0)	19	11	–	11	0.02	0.13	0.15
	(4,4,0)	17	11	–	11	0.04	0.20	0.24
IDCT-c (67)	(3,1,2)	24	25	26	25	0.26	0.62	0.88
	(5,2,4)	17	25	27	25	0.32	0.94	1.26
	(11,9,6)	14	26	31	26	0.55	21.1	21.65
16-FFT (81)	(4,2,0)	17	20	23	20	0.32	112.24	112.56
	(6,2,0)	13	20	25	20	0.41	542.79	543.20
	(16,4,0)	10	16	32	(21)	1.15	N/A	N/A

*(#ALU, #MUL, #SHT)

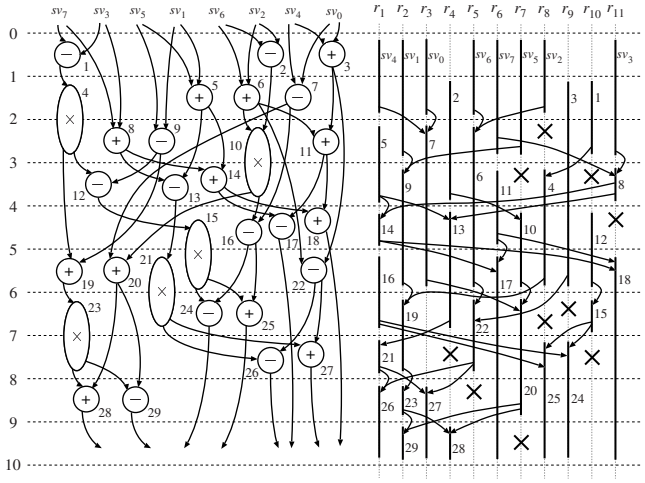

Fig. 9. The result of 8-FFT: the schedule and register assignment for CDD-based design. CDD-order is defined as $r_i \succ_C r_j$ for each $i < j$. Symbols "×" in the right part represent control steps to which any data cannot be assigned.

VII. CONCLUSION

This paper introduced a novel class of datapaths that have robustness against delay variations, which is achieved by CDD-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 that, in all cases except 16-FFT with resource (16,4,0), the CDD-based register assignment achieved their lower bounds in the number of registers. Development of an efficient heuristic algorithm for CDD-based register assignment problem for a large problem instance is one of our future problems. CDD-based register assignment for pipelined datapaths is also our future problem.

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 , and the Telecommunications Advancement Foundation.

APPENDIX: PROOF OF THEOREM 1

In this section, we describe details of the proof of Theorem 1. We have derived a polynomial time reduction from the "modified circular arc coloring problem," which is defined from the circular arc coloring problem, to the decision version of our problem.

The circular arc coloring problem is described as follows, given a set of arcs A on the unit circle, and a positive integer K, determine whether A can be colored with at most K colors. It is well-known that the circular arc coloring problem is NP-complete [11].

Let $A := \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ be a set of arcs on a unit circle $[0, 2\pi)$, where a_i and b_i in (a_i, b_i) are the start

and the end angles (radian), respectively. Note that, without loss of generality, we assume that each arc (a_i, b_i) is open in the sense that neither the point a_i nor the point b_i is included in the arc. We assume that A includes t arcs which include zero between their start and end angles, and without loss of generality, let them be $A_0 := \{(a_1, b_1), (a_2, b_2), \ldots, (a_t, b_t)\}$. Let $a_{\min}$ and $b_{\max}$ be

$$a_{\min} := \min_{t+1 \le i \le n} a_i \text{ and } b_{\max} := \max_{t+1 \le i \le n} b_i,$$

respectively. If $a_{\min} = 0$ and/or $b_{\max} = 2\pi$, we can modify them to $a_{\min} > 0$ and $b_{\max} < 2\pi$ by changing the length of arcs appropriately with keeping overlap/non-overlap between arcs. Let L be a constant defined as follows

$$L := \begin{cases} n & \text{if } K > n; \\ K & \text{otherwise.} \end{cases}$$

Now, we introduce a new set I of arcs defined as follows.

$$\begin{aligned} I := A \cup \\ \{(a_i, b_i) | L > t, a_i = b_{\max}, b_i = a_{\min}, n+1 \le i \le n+L-t\} \\ \cup \{(a_i, b_i) | L \ge t, a_i = 0, b_i = 2\pi, \\ n+L-t+1 \le i \le 2(n+L-t)\} \end{aligned}$$

Fig. 10(a) is an example instance of the circular arc coloring, $A = \{\alpha, \beta, \gamma\}$, $K = 2$. In this example, because $n = 3$, $L = 2$ and $t = 1$, we add one arc $(a_i = b_{\max}, b_i = a_{\min})$ and four round arcs $(a_i = 0, b_i = 2\pi)$. Fig. 10(b) shows the resultant arc set I obtained from A.

Now, we are ready to formulate the following problem.

Problem 1: [The modified circular arc coloring problem] Given a set of arcs A and a positive integer K, determine whether I can be colored with at most $n+2L-t$ colors.

Lemma 1: The modified circular arc coloring problem is NP-complete.

Lemma 1 can be easily shown by reduction from the original circular arc coloring problem. In set I, the number of arcs which include the point zero is $n+2L-t$. Therefore, without loss of optimality, if I is a yes instance of the modified circular arc problem, each data (a_i, b_i), $1 \le i \le t$, $n+1 \le i \le 2(n+L-t)$, can be colored with i. Finally, the remaining data (a_i, b_i), $t+1 \le i \le n$, can be colored at most $n+2L-t$ colors if and only if A can be colored at most K colors.

In the following, we will show the transformation from an instance of the modified circular arc coloring problem to an input instance of the decision version of our CDD-based register assignment problem. Let h_a and h_b be

$$\begin{aligned} h_a &:= a_{\min}/2 \quad \text{and} \\ h_b &:= b_{\max} + (2\pi - b_{\max})/2 = \pi + b_{\max}/2, \end{aligned}$$

respectively. First, a_is, b_is, h_a, and h_b are modified into integers with keeping overlap/non-overlap between arcs.

DFG $G = (\mathcal{O}, \mathcal{A})$: The input DFG is constructed as follows.

$$\begin{aligned} \mathcal{O} := \{O^{(j)}_{0\_\mathrm{div},i} \mid 1 \le i \le t, 2 \le j \le 4\} \\ \cup \{O^{(j)}_{\mathrm{pad},i} \mid 1 \le i \le L-t, 2 \le j \le 4\} \end{aligned}$$

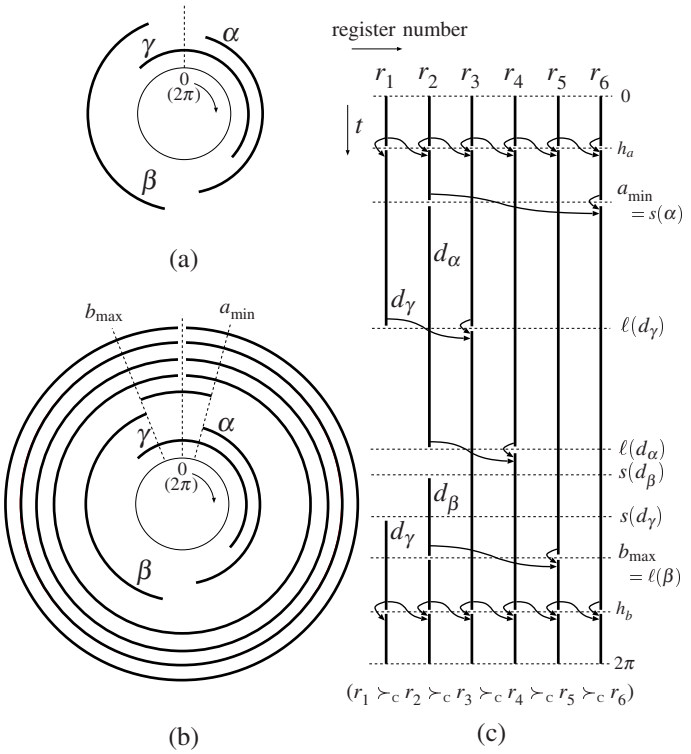

Fig. 10. Example. (a) circular arc set A, $K = 2$. (b) circular arc set I obtained from A. (c) the transformed instance from I and its feasible register assignment.

$$\begin{aligned} \cup \{O^{(j)}_{\mathrm{out\_recv},i} \mid 1 \le i \le n, 2 \le j \le 4\} \\ \cup \{O^{(j)}_{\mathrm{pad\_recv},i} \mid 1 \le i \le L-t, 2 \le j \le 4\} \\ \cup \{O_{\mathrm{others},i} \mid t+1 \le i \le n\} \end{aligned}$$

where all operations in $\mathcal{O}$ are one-cycle add operations which can be executed on a single-type functional unit ALU. We assume that the primary inputs are the outputs of virtual operations $O^{(1)}_{0\_\mathrm{div},i}$, $O^{(1)}_{\mathrm{pad},i}$, $O^{(1)}_{\mathrm{out\_recv},i}$, and $O^{(1)}_{\mathrm{pad\_recv},i}$, which are scheduled to time zero for each i.

$$\begin{aligned} \mathcal{A} := \{(O^{(j)}_{0\_\mathrm{div},i}, O^{(j+1)}_{0\_\mathrm{div},i}) \mid 1 \le i \le t, j = 1,3\} \\ \cup \{(O^{(j)}_{\mathrm{pad},i}, O^{(j+1)}_{\mathrm{pad},i}) \mid 1 \le i \le L-t, j = 1,3\} \\ \cup \{(O^{(j)}_{\mathrm{out\_recv},i}, O^{(j+1)}_{\mathrm{out\_recv},i}) \mid 1 \le i \le n, j = 1,2,3\} \\ \cup \{(O^{(j)}_{\mathrm{pad\_recv},i}, O^{(j+1)}_{\mathrm{pad\_recv},i}) \mid 1 \le i \le L-t, j = 1,2,3\} \\ \cup \{(O^{(j)}_{0\_\mathrm{div},i}, O^{(j+1)}_{0\_\mathrm{div},i+1}) \mid 1 \le i \le t-1, j = 1,3\} \\ \cup \{(O^{(j)}_{\mathrm{pad},i}, O^{(j+1)}_{\mathrm{pad},i+1}) \mid 1 \le i \le L-t-1, j = 1,3\} \\ \cup \{(O^{(j)}_{\mathrm{out\_recv},i}, O^{(j+1)}_{\mathrm{out\_recv},i+1}) \mid 1 \le i \le n-1, j = 1,3\} \\ \cup \{(O^{(j)}_{\mathrm{pad\_recv},i}, O^{(j+1)}_{\mathrm{pad\_recv},i+1}) \mid 1 \le i \le L-t-1, j = 1,3\} \\ \cup \{(O^{(j)}_{0\_\mathrm{div},t}, O^{(j+1)}_{\mathrm{pad},1}) \mid j = 1,3\} \\ \cup \{(O^{(j)}_{\mathrm{pad},L-t}, O^{(j+1)}_{\mathrm{out\_recv},1}) \mid j = 1,3\} \\ \cup \{(O^{(j)}_{\mathrm{out\_recv},n}, O^{(j+1)}_{\mathrm{pad\_recv},1}) \mid j = 1,3\} \\ \cup \{(O^{(2)}_{\mathrm{pad},i}, O^{(3)}_{\mathrm{pad\_recv},i}) \mid 1 \le i \le L-t\} \end{aligned}$$

$$\cup\{(O^{(2)}_{0\_\mathrm{div},i}, O^{(3)}_{\mathrm{out\_recv},i}) \mid 1 \le i \le t\}$$
$$\cup\{(O_{\mathrm{others},i}, O^{(3)}_{\mathrm{out\_recv},i}) \mid t+1 \le i \le n\}$$
$$\cup\{(O^{(3)}_{\mathrm{out\_recv},i}, O^{(3)}_{0\_\mathrm{div},i}) \mid 1 \le i \le t\}$$
$$\cup\{(O^{(2)}_{\mathrm{out\_recv},i}, O_{\mathrm{others},i}) \mid t+1 \le i \le n\}$$
$$\cup\{(O^{(3)}_{\mathrm{pad\_recv},i}, O^{(3)}_{\mathrm{pad},i}) \mid 1 \le i \le L-t\}$$

It is clear that the DFG G defined above is a DAG.

Set of data $\mathcal{D}$: Let the output of $O^{(j)}_{0\_\mathrm{div},i}$, $O^{(j)}_{\mathrm{pad},i}$, $O^{(j)}_{\mathrm{pad\_recv},i}$, $O^{(j)}_{\mathrm{out\_recv},i}$, and $O_{\mathrm{others},i}$ be $d^{(j)}_{0\_\mathrm{div},i}$, $d^{(j)}_{\mathrm{pad},i}$, $d^{(j)}_{\mathrm{pad\_recv},i}$, $d^{(j)}_{\mathrm{out\_recv},i}$, and $d_{\mathrm{others},i}$ for each j, $2 \le j \le 4$, respectively. Let $d^{(1)}_{0\_\mathrm{div},i}$, $d^{(1)}_{\mathrm{pad},i}$, $d^{(1)}_{\mathrm{pad\_recv},i}$, and $d^{(1)}_{\mathrm{out\_recv},i}$ be the primal inputs which are stored in registers from the start, and $d^{(4)}_{0\_\mathrm{div},i}$, $d^{(4)}_{\mathrm{pad},i}$, $d^{(4)}_{\mathrm{pad\_recv},i}$, and $d^{(4)}_{\mathrm{out\_recv},i}$ be the primal outputs which are stored in registers until the end.

Control schedule σ: The input schedule σ is given as follows for each i.

$$\begin{array}{lll}
\sigma(O^{(2)}_{0_\mathrm{div},i}) = h_a, & \sigma(O^{(3)}_{0_\mathrm{div},i}) = a_i, & \sigma(O^{(4)}_{0_\mathrm{div},i}) = h_b, \\
\sigma(O^{(2)}_{\mathrm{pad},i}) = h_a, & \sigma(O^{(3)}_{\mathrm{pad},i}) = b_{\max}, & \sigma(O^{(4)}_{\mathrm{pad},i}) = h_b, \\
\sigma(O^{(2)}_{\mathrm{pad_recv},i}) = h_a, & \sigma(O^{(3)}_{\mathrm{pad_recv},i}) = a_{\min}, & \sigma(O^{(4)}_{\mathrm{pad_recv},i}) = h_b, \\
\sigma(O^{(2)}_{\mathrm{out_recv},i}) = h_a, & \sigma(O^{(3)}_{\mathrm{out_recv},i}) = b_i, & \sigma(O^{(4)}_{\mathrm{out_recv},i}) = h_b, \\
\sigma(O_{\mathrm{others},i}) = a_i &&
\end{array}$$

Sets of resources $\mathcal{F}$ and $\mathcal{R}$: Let $\mathcal{F}$ and $\mathcal{R}$ be the set of ALUs and the set of registers, respectively. The number of resources is defined as $|\mathcal{R}| = |\mathcal{F}| := n + 2L - t$.

FU assignment ρ: From the specification of the control schedule σ, we can see that the number of operations which are assigned to the same control step is at most $|\mathcal{F}|$. Since the lifetimes of operations are considered as the set of intervals, we can apply the left-edge algorithm to compute FU assignment.

Regarding start and last time of each data lifetime as a control step, we can get an instance of our problem. Fig. 10(c) shows an instance transformed from Fig. 10(b), where a horizontal dotted line represents a control step, a vertical line represents a data lifetime, r_i represents a register, and data drawn to the same row mean data assigned the same register. Data d_i corresponds to arc i. The symbol $s(i)$ and $\ell(i)$ is the start and the last time of data-lifetime of d_i, respectively. CDD-order is defined as $r_i \succ_{\mathrm{c}} r_j$, for each $i < j$.

We can have following lemmas. Because a lack of space, the proofs of the following lemmas are omitted.

Lemma 2: If the resultant instance is a yes instance, we have equality $\xi(d^{(j)}_{0\_\mathrm{div},i}) = \xi(d^{(k)}_{0\_\mathrm{div},i})$ for each j, k, $1 \le j,k \le 4$. Similar equality holds for $d^{(j)}_{\mathrm{pad}}$, $d^{(j)}_{\mathrm{pad\_recv}}$, and $d^{(j)}_{\mathrm{out\_recv}}$.

Lemma 3: If the resultant instance is a yes instance, we have to define CDD-order as following order: $\xi(d^{(j)}_{0\_\mathrm{div},1}) \succ_{\mathrm{c}} \cdots \succ_{\mathrm{c}} \xi(d^{(j)}_{0\_\mathrm{div},t}) \succ_{\mathrm{c}} \xi(d^{(j)}_{\mathrm{pad},1}) \succ_{\mathrm{c}} \cdots \succ_{\mathrm{c}} \xi(d^{(j)}_{\mathrm{pad},L-t}) \succ_{\mathrm{c}} \xi(d^{(j)}_{\mathrm{pad\_recv},1}) \succ_{\mathrm{c}} \cdots \succ_{\mathrm{c}} \xi(d^{(j)}_{\mathrm{pad\_recv},L-t}) \succ_{\mathrm{c}} \xi(d^{(j)}_{\mathrm{out\_recv},1}) \succ_{\mathrm{c}} \cdots \succ_{\mathrm{c}} \xi(d^{(j)}_{\mathrm{out\_recv},n})$ for each j, $1 \le j \le 4$.

Proof of Theorem 1: It is easy to see that the decision problem version of our problem is in the class NP. In addition, the transformation from an instance of the circular arc coloring problem to our problem can be done in polynomial time.

First, we assume that I can be colored using at most $n + 2L - t$ colors. Without loss of generality, let the color function c be partially defined as follows: $c(d^{(j)}_{0\_\mathrm{div},i}) := i, c(d^{(j)}_{\mathrm{pad},i}) := t + i, c(d^{(j)}_{\mathrm{pad\_recv},i}) := L + i, c(d^{(j)}_{\mathrm{out\_recv},i}) := i + 2L - t$ for each i, j, $1 \le j \le 4$. And then, we define register assignment for the transformed instance by seeing colors of each arc to the register number. This register assignment is a feasible one for our problem with CDD-order $r_i \succ_{\mathrm{c}} r_j$ for each i, j, $i < j$.

Next, we assume that the transformed instance can be assigned to at most $n + 2L - t$ registers with a feasible CDD-order. By Lemma 2 and 3, we can easily derive an arc coloring from the assignment. ∎

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. Custom Integrated Circuits Conference (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," *Report of MIRAI Project*, 2004.

[3] J. Jung and T. Kim, "Timing variation-aware high-level synthesis," *Proc. International Conference on Computer-Aided Design (ICCAD)*, pp. 424–428, November 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, November 2007.

[5] Z. Wang, M. M-Sadowska, K-H. Tsai, and J. Rajski, "Diagnosis of hold time defects," *Proc. International Conference on Computer Design (ICCD)*, pp. 192–199, October 2004.

[6] J. Xi and W W.-M. Dai, "Jitter-tolerant clock routing in two-phase synchronous systems," *Proc. International Conference on Computer-Aided Design (ICCAD)*, pp. 316–320, November 1996.

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

[8] 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.

[9] 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.

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

[11] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou, "The complexity of coloring circular arcs and chords," *SIAM Journal on Algebraic Discrete Methods*, vol. 1, no. 2, pp. 216–227, 1980.

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

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

[14] R. R. Hoare, A. K. Jones, D. Kusic, J. Fazekas, J. Foster, S. Tung, and M. McCloud, "Rapid VLIW processor customization for signal processing applications using combinational hardware functions," *Proc. EURASIP J. Appl. Signal Process.*, vol. 2006, issue 1, pp. 1–23, January 2008.

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