PAPER Special Section on VLSI Design and CAD Algorithms

Simultaneous Optimization of Skew and Control Step Assignments in RT-Datapath Synthesis

Takayuki OBATA[†] *and* **Mineo KANEKO**[†a)]**,** ***Members***

SUMMARY As well as the schedule affects system performance, the control skew, i.e., the arrival time difference of control signals between registers, can be utilized for improving the system performance, enhancing robustness against delay variations, etc. The simultaneous optimization of the control step assignment and the control skew assignment is more powerful technique in improving performance. In this paper, firstly, we prove that, even if the execution sequence of operations which are assigned to the same resource is fixed, the simultaneous optimization problem under a fixed clock period is $\mathcal{NP}$-hard. Secondly, we propose a heuristic algorithm for the simultaneous control step and skew optimization under given clock period, and we show how much the simultaneous optimization improves system performance. This paper is the first one that uses the intentional skew to shorten control steps under a specified clock period. The proposed algorithm has the potential to play a central role in various scenarios of skew-aware high level synthesis.

key words: *high level synthesis, RT datapath, skew, wiring delay, scheduling*

1. Introduction

In the logic level VLSI design, the clock skew is now utilized intentionally for improving system performances, enhancing the robustness against delay variations, reducing maximum peak power, etc., and significant efforts have been devoted to so-called clock-scheduling and simultaneous optimization of re-timing and clock-scheduling [1]–[6]. Recently, the importance and the impact of considering timing skew in the high level synthesis are recognized, and researches on skew-aware high level synthesis have been started [7]–[11].

Because of the presence of several different approaches to the high level synthesis, the way of introducing intentional skew (intentionally controlled timing difference between the beginning of a control step and the working timing of each register and multiplexer) into high level synthesis is not unique. One possible scenario is that a conventional synthesis system incorporates intentional timing skew, and uses it to compensate the imbalance of function delays. One other possible scenario is that a concurrent datapath/floorplan synthesis system [12]–[19] incorporates intentional skew, and uses it to compensate imbalance of path delays (each path delay may include function delays and signal propagation delays). Similar to the clock schedule in the logic level design, the skew-aware high level design will contribute to reducing the clock period, and enhancing the robustness against delay variations. Furthermore, it will be demonstrated in this paper that the intentional skew will also contribute to reducing the number of control steps (makespan) for a target application.

It is well-known in the logic level design that the clock skew only is not enough for the highest performance, and the combination of the clock skew with the re-timing technique is a promising approach. Similar to this situation, in the skew-aware high level synthesis, the simultaneous optimization of the control step assignment and the skew assignment has a higher potential in performance optimization.

To discuss mathematically and logically the essential difference between skew and re-timing in a sequential circuit and our skew and control step assignment would be a hard task. The rather superficial difference between two are as follows.

1. Every register reads its input at every clock cycle in a sequential circuit. On the other hand, each register reads its input only scheduled clock cycle.
2. By re-timing technique in a sequential circuit, delay between registers will change, but still every register keeps to read its input at every clock cycle. On the other hand, in a datapath circuit, control step assignment (re-scheduling) does not change any delay between registers, but changes the timing in which each register reads its input.
3. Skew and re-timing for a sequential circuit target on reducing clock period. On the other hand, our skew and re-schedule can work not only for reducing clock period, but also for reducing schedule length. Furthermore, it is not clear which value (or concept) in a sequential circuit corresponds to the schedule length and varies by re-timing.

So, as a result, skew and re-timing algorithms designed so far for a sequential circuit can not be applied to our problem, especially for reducing schedule length.

Taking the peculiarity of the skew assignment into consideration, we assume that resource binding and the temporal order (not a specific control step assignment) of lifetimes of data assigned to the same register are fixed as a part of input description to our problem, and we try to optimize skew and control step assignments under those constraints. It is expected that, if we have a tool to solve this problem, it can be used also as a sub-tool for optimizing resource binding

Manuscript received March 17, 2008.
Manuscript revised June 24, 2008.

[†]The authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi-shi, 923-1292 Japan.
a) E-mail: mkaneko@jaist.ac.jp

DOI: 10.1093/ietfec/e91–a.12.3585

and temporal order of lifetimes.

The first contribution of this paper is to show that our simultaneous control-step and skew optimization is a NP-hard problem, whereas the skew optimization under fixed control-step assignment and the constrained control-step optimization (that is, resource binding and the temporal order of lifetimes of data assigned to the same register are fixed) under fixed skew assignment are in the class P. The second contribution of this paper is a heuristic algorithm for our simultaneous control-step and skew optimization problem. In the past, the intentional skew was used mainly to shorten a clock period, and as a result, the clock period was not controlled intentionally. This paper is the first one that uses the intentional skew to shorten control steps under a specified clock period.

This paper is organized as follows. In Sect. 2, we summarize basic notations, and show motivational example. In Sect. 3, our simultaneous optimization of the control step assignment and the skew assignment is formulated. Section 4 deals with the computational complexity of our problem. We present a heuristic algorithm in Sect. 5. Experimental results are shown in Sect. 6. Finally, we present conclusions in Sect. 7.

2. Background and Motivation

2.1 Structural and Behavioral Descriptions of Datapath Circuit

We assume that the input algorithm of high-level synthesis is described as a data flow graph (DFG in short) $(\mathcal{O}, \mathcal{D})$ where a vertex set $\mathcal{O}$ is the set of operations and an edge set $\mathcal{D}$ indicates data dependencies between operations.

The input algorithm is transformed to the datapath circuit by determining resource assignment, that is, the functional unit assignment $\rho : \mathcal{O} \to \mathcal{F}$ and the register assignment $\xi : \mathcal{O} \setminus \mathcal{U} \to \mathcal{R}$, where $\mathcal{F}$ is a set of functional units, $\mathcal{R}$ is a set of registers, $\mathcal{U}$ is a set of operations whose outputs are not written to registers, and $\xi(o) = r$ means that the output data of the operation o is assigned to a register r (the output of o is written in r). Interconnections and multiplexers in the datapath part are so designed that, for each operation o_j with $(o_i, o_j) \in \mathcal{D}$, the input register $\xi(o_i)$ is connected to the functional unit $\rho(o_j)$ and $\rho(o_j)$ is connected to the output register $\xi(o_j)$. As an example, Fig. 1(b) shows the datapath circuit obtained from Fig. 1(a) with the resource assignment $\rho(o1) = FU1, \rho(o2) = FU2, \rho(o3) = FU2, \rho(o4) = FU1, \rho(o5) = FU1, \xi(o1) = r2, \xi(o2) = r5, \xi(o3) = r3, \xi(o4) = r4, \xi(o5) = r1$.

The behavior of the datapath circuit, on the other hand, is determined by the arrival timing of control signals to registers and multiplexers. We let $\mathcal{M}$ be the set of all registers (and multiplexers), and $\mathcal{S}$ denotes the set of all control signals, where $c_x^o \in \mathcal{S}$ represents the control signal which is related to the execution of $o \in \mathcal{O}$ and is sent to $x \in \mathcal{M}$. The arrival timing is partly determined by the control step assignment $\sigma : \mathcal{S} \to \mathbb{Z}_+$, and the rest by the timing skew $\tau : \mathcal{M} \to \mathbb{R}$. As the result, the control signal c_x^o reaches x at the time $\sigma(c_x^o) \cdot clk + \tau(x)$, where clk is a clock period.

2.2 Motivational Example

Figure 2(a) shows an example of a DFG. We assume that the resource binding has been finished, and for each operation, signal path delays from an input register to an output register via a functional unit has been obtained. In general, data path from a multiple-bit register to a multiple-bit register must be

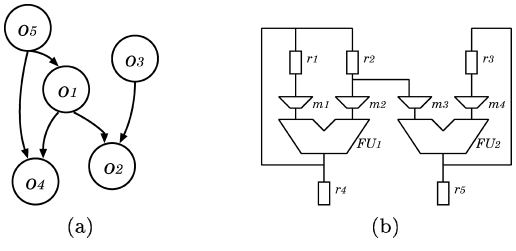

Fig. 1 Example of DFG and RT-Level architecture.

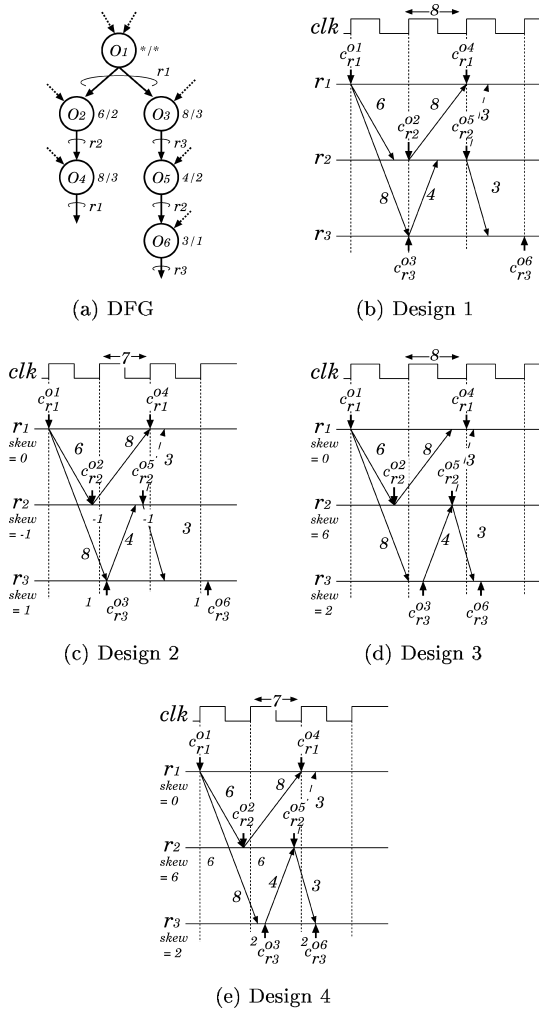

Fig. 2 Several different types of skew aware designs.

a multiple-input, multiple-output combinatorial circuit, and hence data path from a register to a register includes multiple signal paths having different delays. We will characterize each register-to-register data path with the maximum and the minimum among those delays, and we call them the maximum delay and the minimum delay, respectively. In Fig. 2(a), the maximum and the minimum delays are indicated like 6/2 for O_2, 8/3 for O_3, etc. The assignment of data to registers is indicated as r_i beside each arc.

Figure 2(b) shows a schedule of 6 control signals $c^{o_1}_{r_1}$, $c^{o_2}_{r_2}$, $c^{o_3}_{r_3}$, $c^{o_4}_{r_1}$, $c^{o_5}_{r_2}$, and $c^{o_6}_{r_3}$. The number written beside a slant solid (broken) arrow shows maximum (minimum) delay of the corresponding operation. The schedule requires 4 control steps, and its minimum clock period is 8 (the total computation time is $8 \times 4 = 32$). This is an optimum schedule under zero skew if the number of control steps is restricted to smaller than or equal to 4 (See Fig. 2(b)). When we assign skew $(\tau(r1), \tau(r2), \tau(r3)) = (0, -1, 1)$, the minimum clock period can be reduced to 7 (the total computation time is now $7 \times 4 + 1 = 29$). The situation is illustrated in Fig. 2(c). When we assign skew $(\tau(r1), \tau(r2), \tau(r3)) = (0, 6, 2)$ and we keep clock period to 8, we can modify the schedule and the number of control steps can be reduced to 3 (totally, $8 \times 3 + 2 = 26$). The situation is illustrated in Fig. 2(d). Figure 2(e) shows an optimum schedule and skew assignment. If we try to keep the number of control steps to 3, the minimum clock period is now 7 (totally, $7 \times 3 + 2 = 23$).

Skew is conventionally utilized only for reducing clock period (Fig. 2(c)). However, in many design flows, the clock period can not be determined freely upon convenience of a single datapath circuit. If we need to design a datapath under a given clock period, the conventional skew optimization can not be used directly. As it is shown by Design 3 in Fig. 2(d), the simultaneous optimization of the skew assignment and the control step assignment is necessary for datapath synthesis under a given clock period. Also in the case for minimizing the latency (Design 4 in Fig. 2(e)), the skew assignment and the control step assignment must be treated simultaneously.

3. Simultaneous Optimization of Control Step and Skew Assignments

3.1 Formulation of the Problem

Our simultaneous optimization problem, (σ, τ, clk)-optimization, receives a data flow graph G, resource assignments ρ, ξ, and the execution order of operations assigned to the same FU, and the production order of data assigned to the same register (the function *next* introduced later reflects such information), and outputs σ, τ and clk.

Figure 3 illustrates the correct timing of control signals with respect to the execution of o_j. We assume that o_i is an operation generating an input of o_j, and the output of the operation o_j is written in a register r_j ($\xi(o_j) = r_j$). On the other hand, the resource x_i is either a register which stores the input data for o_j, an input multiplexer of a FU $\rho(o_j)$, or an input multiplexer of r_j.

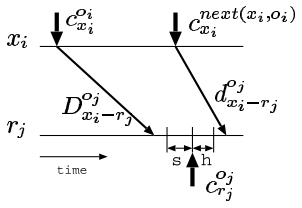

Fig. 3 Setup/Hold constants.

The "setup constraint" (the arrival of the control signal $c^{o_j}_{r_j}$ has to be later than the arrival of the result of o_j) is formulated as

$$\sigma(c^{o_*}_{x_i}) \cdot clk + \tau(x_i) + t_{err} + D^{o_j}_{x_i-r_j} + s \leq \sigma(c^{o_j}_{r_j}) \cdot clk + \tau(r_j) \quad (1)$$

where o_* is either the operation that generates the input of o_j stored in a register x_i or o_j in case x_i is a multiplexer at the input of $\rho(o_j)$ or r_j. $D^{o_j}_{x_i-r_j}$ is the maximum path delay from x_i to r_j related to the execution of o_j. t_{err} is a timing margin, and s is the setup time of the register r_j.

On the other hand, the "hold constraint" (the arrival of $c^{o_j}_{r_j}$ has to be earlier than the destruction of the result of o_j) is given as

$$\sigma(c^{o_j}_{r_j}) \cdot clk + \tau(r_j) + t_{err} \leq \sigma(c^{next(x_i,o_i)}_{x_i}) \cdot clk + \tau(x_i) + d^{o_j}_{x_i-r_j} - h, \quad (2)$$

where $d^{o_j}_{x_i-r_j}$ is the minimum path delay from x_i to r_j related to the execution of o_j, and h is the hold time of r_j. $next(x_i, o_i)$ is the operation next to o_i on the resource x_i. In case that several operations are chained, setup and hold constraints are formulated between input registers to the first operation, intermediate multiplexers located on the chaining path, and the output register of the last operation of the chain. Note that, when the control step assignment σ is variable, we can set $0 \leq \tau(x) < clk$ for all $x \in \mathcal{M}$ without loss of optimality.

Note that schedule and skew for an input multiplexer of a register is trivial. Let m be an input multiplexer of a register r. For all operations o which are assigned to the same register r, the maximum path delays D^o_{m-r}s have almost the same value, and also the minimum path delays d^o_{m-r}s have almost the same value. So we can set $\sigma(c^o_m)$ and $\tau(m)$ as following.

$$\sigma(c^o_m) = \left\lfloor \frac{\sigma(c^o_r) \cdot clk + \tau(r) - D^o_{m-r}}{clk} \right\rfloor,$$

$$\tau(m) = \tau(r) - D^o_{m-r} - \left\lfloor \frac{\tau(r) - D^o_{m-r}}{clk} \right\rfloor clk.$$

In general, the objective of the scheduling is the minimization of the computation time and the size of a resultant circuit. Since ξ and ρ are fixed in our problem, to minimize the size of a circuit is to minimize the size of a controller. We can assume that the size of the controller is an increasing

function of $|\mathcal{M}|$ and CS (the number of control steps). Since $|\mathcal{M}|$ is fixed, CS is our objective to be minimized in terms of circuit size. On the other hand, the computation time of a circuit can be evaluated with $clk \cdot CS$. Hence, we choose $clk \cdot CS + \lambda \cdot CS$ as the objective to be minimized, where λ is a weighting coefficient.

Whether an instance has a feasible solution or not can be tested easily by relaxing integer-valued problem into real-valued problem, that is, by relaxing $\sigma : S \rightarrow Z_+$ into $\sigma : S \rightarrow R_{\geq 0}$. We will call $\sigma : S \rightarrow R_{\geq 0}$ a real-valued schedule and it is computed by using a real-domain schedule constraint graph $G_{rs} = (\mathcal{S}, E_{rs})$. The vertex set $\mathcal{S}$ is the set of all control signals. The weighted edge set E_{rs} corresponds to the following constraint inequalities obtained from (1)–(2) assuming variables τ equal to zero.

$$\sigma(c_{r_j}^{o_j}) \geq \sigma(c_{x_i}^{o_i}) + D_{x_i - r_j}^{o_j} + s + t_{err} \tag{3}$$

$$\sigma(c_{x_i}^{next(x_i,o_i)}) \geq \sigma(c_{r_j}^{o_j}) - d_{x_i - r_j}^{o_j} + h + t_{err} \tag{4}$$

That is, corresponding to (3)–(4), we add $(c_{x_i}^{o_i}, c_{r_j}^{o_j})$ with its weight $D_{x_i - r_j}^{o_j} + s + t_{err}$ and $(c_{r_j}^{o_j}, c_{x_i}^{next(x_i,o_i)})$ with its weight $-d_{x_i - r_j}^{o_j} + h + t_{err}$. The longest path length to each vertex from a source in G_{rs} provides a feasible and optimal (in terms of $clk \times CS \rightarrow min$) real-valued schedule.

Theorem 1. *If and only if there is no positive cycle in G_{rs}, there is a feasible assignment of σ and τ.*

If there is no feasible real-valued schedule, there is no feasible assignment of σ and τ because if there is a feasible assignment of σ and τ, we can compute the actual time of control signals in the form of $\sigma(c_x^o) \cdot clk + \tau(x)$. When clk is sufficiently small, τ is also small, nearly equal to zero, and we can assume that $\sigma(c_x^o) \cdot clk$ takes an arbitrary real value. That is a reason why if there is a feasible real-valued schedule, there is a feasible assignment of σ and τ.

σ takes an integer value and τ takes a real value, so if one of clk and CS is fixed, the problem becomes a Mixed Integer Linear Programming Problem. We show an optimization algorithm to compute optimum σ and τ using MILP solver in Fig. 4. Although the algorithm computes an optimum solution, it takes impractical time.

Input λ, constraints.
Step1 Set the objective of MILP to the minimization of CS. Solve the MILP and let CS_{min} be the minimum CS.
Step2 Set clk to 1 and the objective to the minimization of CS. Relax the problem into a real valued LP problem, and solve it. Let $time_{min}$ be the minimum CS
Step3 Set opt to ∞ and CS to CS_{min}.
Step4 Set the objective to the minimization of clk and solve MILP. If $clk \cdot CS + \lambda \cdot CS < opt$ then set opt to $(clk \cdot CS + \lambda \cdot CS)$, CS_{opt} to CS and sol to (σ, τ).
Step5 Increase CS by 1. If $(clk \cdot CS - time_{min}) > \lambda$ then goto Step4, otherwise output sol and terminate.

Fig. 4 A simultaneous optimization algorithm using MILP.

3.2 Partial Problems

Because simultaneous optimization of σ, τ, clk is a hard problem, we consider partial problems. We assume that one of σ, τ, clk is fixed, and try to optimize the other two.

(τ, clk)-optimization is the problem to optimize τ and clk while keeping control schedule σ unchanged. Because τ and clk take real values, (τ, clk)-optimization problem can be formulated as LP problem. An efficient graph theoretic approach has been proposed [9].

(σ, clk)-optimization is the problem to optimize σ and clk under given skew τ. Conventional high level synthesis systems treat (σ, clk)-optimization with zero skew.

(σ, τ)-optimization is the problem to optimize σ and τ under a give clock period clk. In most cases clk may be determined considering various factors, and we often encounter this type of optimization problem. (σ, τ)-optimization is also a good candidate subroutine for solving the original (σ, τ, clk)-optimization. That is, by repeating (σ, τ)-optimization with a systematic sweep of clk, we can find a best solution for (σ, τ, clk)-optimization. Thus, we discuss about σ and τ optimization under given clk in the rest of this paper.

4. Computational Complexity of (σ, τ)-Optimization Problem

(σ, τ)-optimization problem itself is an important problem which we will encounter in various design styles. It may also play an important role in solving the original (σ, τ, clk)-optimization as a powerful subroutine.

The following theorem is one highlight of this paper.

Theorem 2. *(σ, τ)-optimization problem is NP-hard.*

As it is mentioned in Sect. 3.1, our (σ, τ)-optimization problem receives resource assignment and operation order on each FU as a part of input instance. It can be easily seen that, if the skew τ is fixed a priori (zero skew is one choice), σ-optimization problem with given resource assignment and given operation order on each FU is in class P. In contrast, even if resource assignment and operation order on each FU are specified, our (σ, τ)-optimization is a NP-hard problem.

This result would be important in the sense that it shows us a broad and crucial guideline for designing a solution algorithm. That is, the result implies that there is no polynomial time algorithm for (σ, τ)-optimization problem unless P = NP. It also suggests that we need to design a heuristic algorithm, otherwise we have to use an algorithm that is essentially the same with full search.

Based on this result, we will present a heuristic algorithm for (σ, τ)-optimization in the next section. As it is the case for most NP-hard problems, the proof of NP-hardness do not have any direct relation to our heuristic algorithm.

Our proof of Theorem 2 is based on the polynomial time reduction from 3SAT to the decision version of (σ, τ)-optimization problem. The detailed proof is presented in

Appendix.

5. Heuristic Algorithm

In this section, we show a heuristic algorithm for this (σ, τ)-optimization problem.

5.1 Skew Constraint Graph

From (1) and (2), we have

$$\tau_r - \tau_m \geq (\sigma_m^{o_p} - \sigma_r^{o_k}) \cdot clk + \tau_{err} + D_{m-r}^{o_k} + s, \quad (5)$$

$$\tau_m - \tau_r \geq (\sigma_r^{o_k} - \sigma_m^{next(m,o_p)}) \cdot clk + \tau_{err} - d_{m-r}^{o_k} + h. \quad (6)$$

We generate a skew constraint multigraph $G_\tau = (V, E)$ from (5) and (6) as shown in Fig. 5. V is a set of multiplexers, registers and one auxiliary source node v_s. A set of weighted edges E is the union of a set of edges reflecting (5) or (6) (i.e., an edge (m, r) with its weight $(\sigma_m^{o_p} - \sigma_r^{o_k}) \cdot clk + \tau_{err} + D_{m-r}^{o_k} + s$ or an edge (r, m) with its weight $(\sigma_r^{o_k} - \sigma_m^{next(m,o_p)}) \cdot clk + \tau_{err} - d_{m-r}^{o_k} + h$ over all operations), and a set of auxiliary edges $\{(m, v_s) | m \in V \backslash v_s\} \cup \{(v_s, m) | m \in V \backslash v_s\}$. Edge weights for $\{(m, v_s) | m \in V \backslash v_s\}$ and $\{(v_s, m) | m \in V \backslash v_s\}$ are $-clk$ and 0, respectively. Then, skew assignment problem is now considered as the problem to assign real values to vertices in G_τ. Once σ is fixed, maximum path lengths from v_s to other vertices give us a solution, i.e., skews of registers and multiplexers. If G_τ has a positive cycle, feasible skew schedule does not exist.

5.2 Schedule Constraint Graph

From (1) and (2) with regarding integral σ, we have

$$\sigma(c_{r_j}^{o_j}) - \sigma(c_{x_i}^{o_i}) \geq \left\lceil \left(\tau(x_i) - \tau(r_j) + t_{err} + D_{x_i - r_j}^{o_j} + s \right) / clk \right\rceil, \quad (7)$$

$$\sigma(c_{x_i}^{next(x_i,o_i)}) - \sigma(c_{r_j}^{o_j}) \geq \left\lceil \left(\tau(r_j) - \tau(x_i) + t_{err} - d_{x_i - r_j}^{o_j} + h \right) / clk \right\rceil \quad (8)$$

We generate a schedule constraint graph $G_\sigma = (V_\sigma, E_\sigma)$ similar to a skew constraint graph. $V_\sigma = \mathcal{S} \bigcup \{v_s\}$ where v_s is an auxiliary source node. E_σ is the set of edges reflecting (7) or (8), and (v_s, v) for all $v \in \mathcal{S}$ whose weight is 0. Once τ and clk are given, the longest path length from v_s to each node v is a feasible value of $\sigma(v)$, and the maximum of those longest path lengths gives us CS. A path which gives CS is called a critical path. If G_σ has a positive cycle, feasible schedule does not exist.

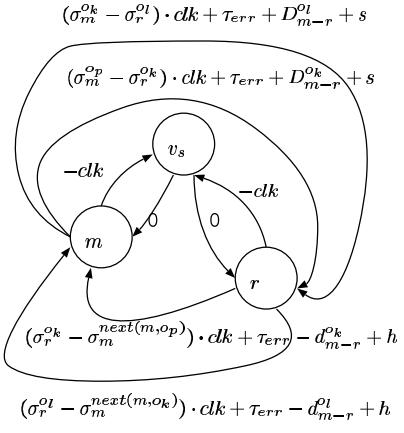

Fig. 5 Skew constraint multigraph.

5.3 Heuristic Algorithm for (σ, τ)-Optimization Problem

Suppose we have computed τ from G_τ, and consider the union T of a longest path from v_s to each node. Then, T is a spanning tree, and for each edge (x_i, r_j) in T, relative skew $(\tau(r_j) - \tau(x_i)) \bmod clk$ is equal to either "$(t_{err} + D_{x_i - r_j}^{o_j} + s) \bmod clk$" or "$(t_{err} - d_{x_i - r_j}^{o_j} + h) \bmod clk$" depending on the edge weight. Therefore, we can consider the skew optimization problem as the problem to extract a spanning tree from G_τ. It is interesting that, once a spanning tree T of G_τ is fixed, and tree edges are suppose to be critical path edges in G_τ, we can compute τ from $T \subset G_\tau$ without information of σ. That is, for each edge (x_i, r_j) in T, we can put the relative-skew $(\tau(r_j) - \tau(x_i)) \bmod clk$ as either $(t_{err} + D_{x_i - r_j}^{o_j} + s) \bmod clk$ or $(t_{err} - d_{x_i - r_j}^{o_j} + h) \bmod clk$ depending on the edge weight.

On the other hand in G_σ, since the right hand side of (7), (8) has a ceiling, if the relative skew $(\tau(r_j) - \tau(x_i)) \bmod clk$ is equal to $(t_{err} + D_{x_i - r_j}^{o_j} + s) \bmod clk$ or $(t_{err} - d_{x_i - r_j}^{o_j} + h) \bmod clk$, the weight of the edge reflecting inequality (7) or (8) is minimized. Therefore, to minimize CS, it is efficient to set relative skew to $(t_{err} + D_{x_i - r_j}^{o_j} + s) \bmod clk$ or $(t_{err} - d_{x_i - r_j}^{o_j} + h) \bmod clk$ for as many edges as possible in a critical path. That is, we have to choose as many edges in a critical path in G_σ as edges of spanning tree in G_τ.

Our heuristic algorithm is shown in Fig. 7. We start with the spanning tree T whose edge set is $\{(v_s, m) | m \in V \backslash v_s\}$ assuming $\tau(m) = 0$ for all m. We replace (v_s, m) with an edge corresponding to an edge on a critical path in G_σ in one by one manner. In each time, we will test all candidate edges to be added to T, and choose one edge which achieves smallest CS.

In order to keep T being a tree, we use a partitioning to know which edge we can add and which edge we have to remove. Each connected component in $T \backslash \{(v_s, x) | x \in V\}$ forms a partite set of a partition. Because T is a tree, only

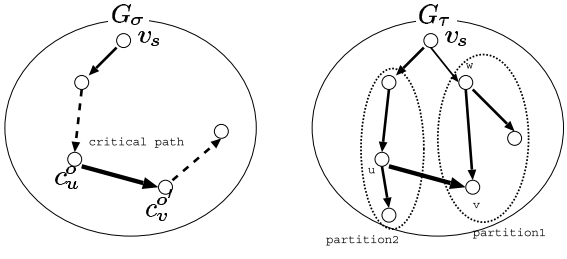

Fig. 6 We add an edge on a critical path in G_σ to T.

Step1. Generate G_τ and G_σ

Step2. Generate an initial spanning tree $T \subset G_\tau$.

Step3. Compute τ_T from T. Compute σ from $G_\sigma|_{\tau=\tau_t}$. Let $\mathcal{P}$ be a critical path in $G_\sigma|_{\tau=\tau_t}$.

Step5. For each edge $(u, v) \in G_\tau$ corresponding to $e \in \mathcal{P}$, try to generate $T_{(u,v)}$ from T by adding (u, v) and removing an appropriate edge. If we can compute $T_{(u,v)}$, compute skew assignment $\tau_{(u,v)}$ from $T_{(u,v)}$ and the number of control steps $CS_{(u,v)}$ from $G_\sigma|_{\tau=\tau_t}$.

Step6. If $CS_{(u,v)} > CS$ or we cannot generate $T_{(u,v)}$ for all (u, v) in Step5, output τ_T and σ and quit. Otherwise, set $T = T_{(u,v)}$ by such (u, v) which achieves the smallest $CS_{(u,v)}$, and go to Step3.

Fig. 7 Heuristic algorithm.

one edge from each partite set connects to v_s. If we generate $T_{(u,v)}$ by adding (u, v) to T, we remove the edge between v_s and the connected component to which v belongs to. G_τ in Fig. 6 shows the replacement of edges. We add (u, v) to T only if u and v belong to different partite sets.

6. Experiments

The proposed heuristic algorithm for (σ, τ)-optimization has been implemented using C programming language and tested on AMD Opteron^{TM} based PC. As input applications, we use three DAG algorithms modified from Jaumann wave filter, all-pole lattice filter and elliptic wave filter.

We have prepared 2 input instances for each input algorithm, each instance has different resource assignment, different operation order, and different delay assignment.

A path delay from an input register to an output register is the sum of delays of register-to-multiplexer, multiplexer-to-FU, FU, FU-to-multiplexer, and multiplexer-to-register. Maximum/minimum delays of multipliers and adders are 60/10 and 20/10, respectively. The other delays are given randomly. That is, minimum register-multiplexer and FU-multiplexer delays are chosen between 3 to 25, and the minimum multiplexer-register and multiplexer-FU delays are chosen from 2 to 15. The maximum delay of each path is set as 1.1 to 1.4 times larger value than its minimum delay.

Note that, for each input instance, those maximum delay values and minimum delay values, as well as resource assignment and operation order on each FU, are fixed throughout the experiments done with various different clock period *clk*. Of course, skew and control step are determined so that the setup condition and the hold condition are satisfied for all operations under the specified maximum delays, minimum delays, and clock period.

As the first experiment, for each instance, we have applied a schedule optimization without skew optimization (assuming zero skew) and the proposed algorithm.

Table 1 shows some of experimental results. The columns "#fu," "#reg," and "*clk*" represent the numbers of functional units, registers, and clock period of each instance, respectively. The column "CS" represents the number of control steps (makespan) of an output schedule and "time" represents the computation time in milli seconds.

Table 1 Experimental results.

Instance	#fu	#reg	*clk*	CS		time(ms)	
				n/s	w/s	n/s	w/s
Jaumann1	6	6	20	38	33	0.122	8.31
			40	22	18	0.124	11.4
			60	18	13	0.125	12.4
			80	14	11	0.123	11.9
			100	13	11	0.122	14.9
Jaumann2	7	7	20	33	31	0.114	1.72
			40	19	17	0.114	3.05
			60	13	12	0.113	11.3
			80	11	9	0.115	10.5
			100	11	9	0.116	9.13
Lattice1	3	5	20	55	50	0.075	2.12
			40	31	27	0.076	3.05
			60	24	18	0.080	5.37
			80	19	15	0.075	3.80
			100	15	12	0.073	5.64
Lattice2	4	5	20	50	46	0.078	2.76
			40	29	25	0.078	3.33
			60	19	16	0.078	3.43
			80	17	14	0.077	5.02
			100	16	13	0.084	6.10
Elliptic1	8	13	20	66	57	0.241	42.1
			40	38	33	0.241	74.0
			60	32	24	0.238	88.6
			80	23	16	0.239	96.5
			100	19	15	0.232	108
Elliptic2	8	14	20	67	58	0.245	42.2
			40	38	31	0.244	51.6
			60	32	20	0.240	57.8
			80	22	18	0.243	101
			100	20	17	0.242	90.6

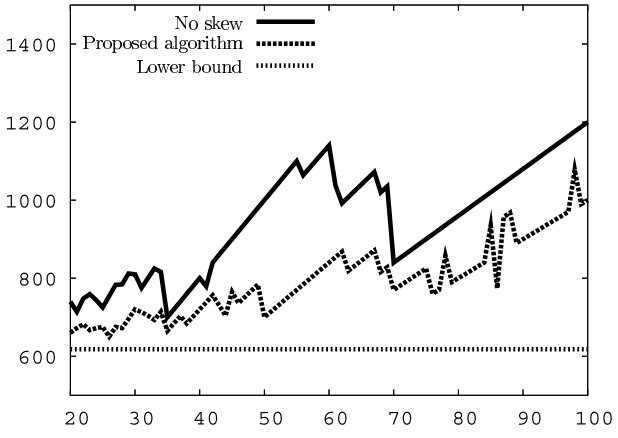

Fig. 8 Application time ($CS \times clk$) vs. *clk* for Jaumann.

Figures 8 through 10 show the experimental results graphically by plotting design points on the application time (i.e., $CS \times clk$) vs. clock period axes. Those plots are obtained by applying our algorithm repeatedly with increasing *clk* by 1 at a time. Note again that, for each input instance, resource assignment, operation order on each FU, maximum delay values and minimum delay values are fixed, and these same values are used for different *clk* values. Lower bound in each figure represents the minimum $CS \times clk$ of real-

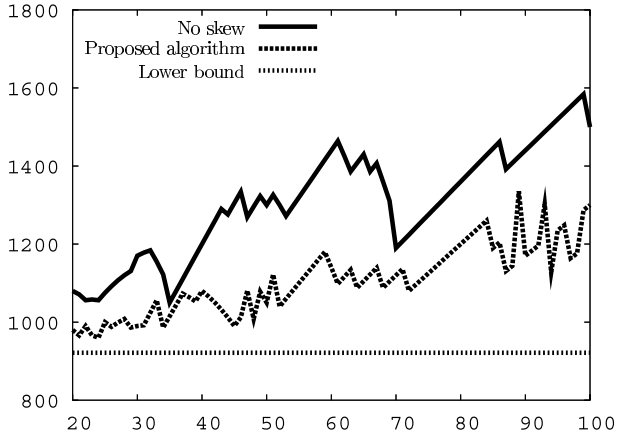

Fig. 9 Application time ($CS \times clk$) vs. clk for Lattice.

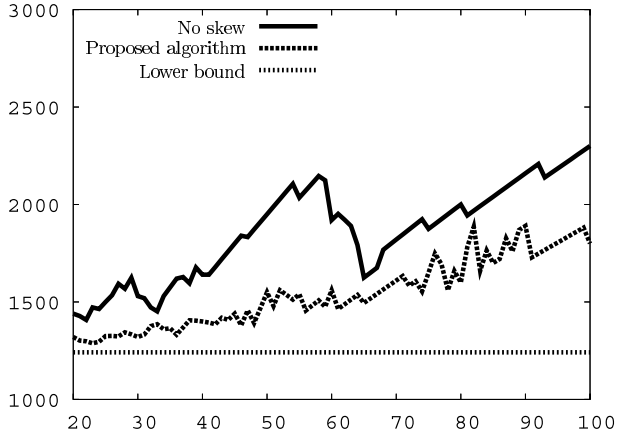

Fig. 10 Application time ($CS \times clk$) vs. clk for Elliptic.

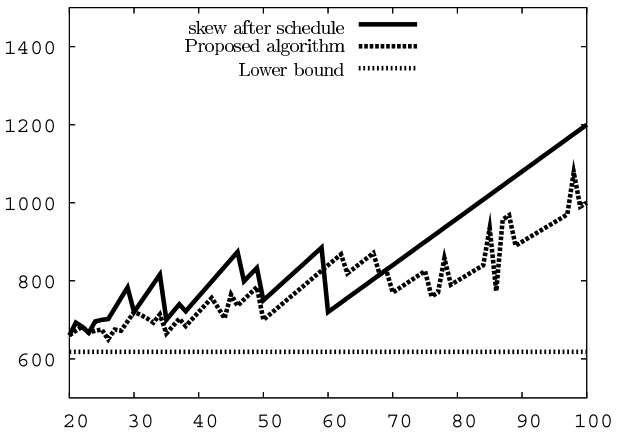

Fig. 11 Application time ($CS \times clk$) vs. clk for Jaumann.

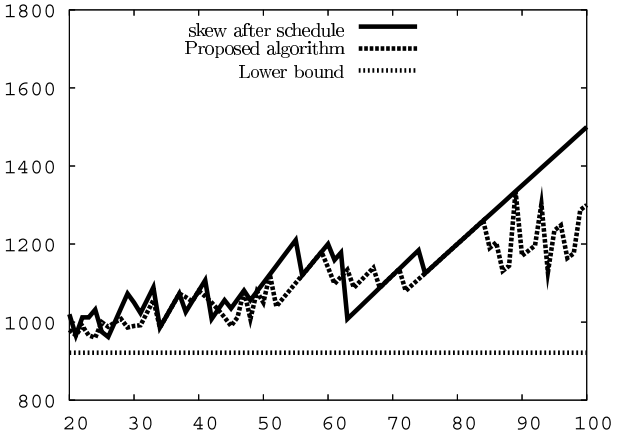

Fig. 12 Application time ($CS \times clk$) vs. clk for Lattice.

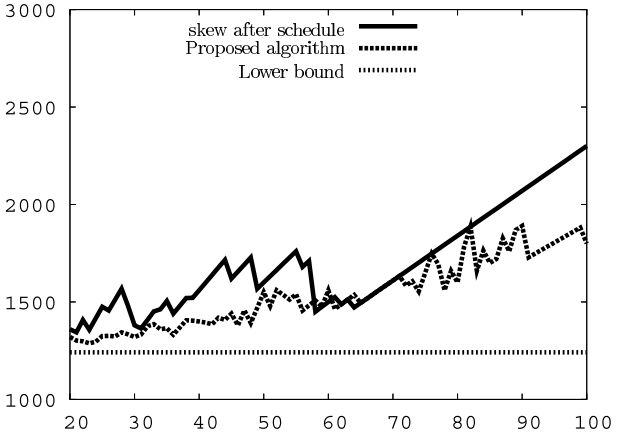

Fig. 13 Application time ($CS \times clk$) vs. clk for Elliptic.

valued schedule, which is introduced in Sect. 3.1. Since the solution space for real-valued schedule includes the one for integer-valued schedule (with skew), the smallest $CS \times clk$ achieved by real-valued schedule is no larger than the smallest $CS \times clk$ achieved by integer-valued schedule with skew.

As it is mentioned previously, conventional skew optimization algorithm is designed only for reducing the clock period. Even though its objective does not match with our objective; reduce the schedule length CS under a given clock period, we will bravely compare our method with a two-step method; scheduling followed by skew optimization. Results are shown in Figs. 11 through 16. Design points given by "skew after schedule" are the result of the two-step algorithm; scheduling (with zero skew) followed by skew optimization. Figures 11, 12, and 13 compare our proposed algorithm with "skew after schedule." On the other hand, "skew after proposed" is also a two-step algorithm; proposed algorithm followed by skew optimization for reducing clock period. Figures 14, 15, and 16 compare "skew after proposed" with "skew after schedule."

From those experiments, advantage of our proposed method to the conventional skew optimization can be verified It is notable that the advantage is remarkable especially when we choose a large clock period (low clock frequency).

Finally, we briefly discuss area overhead and extra power consumption paid for skew control. The following discussion is based on logic synthesis results done by "Design Compiler (Version A-2007, 12-SP3)" with using library "class." Table 2 shows the synthesis results of datapath components, and Table 3 shows the synthesis result of delay el-

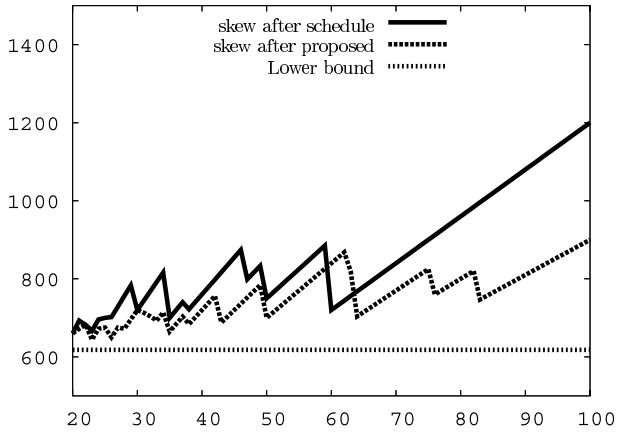

Fig. 14 Application time ($CS \times clk$) vs. clk for Jaumann.

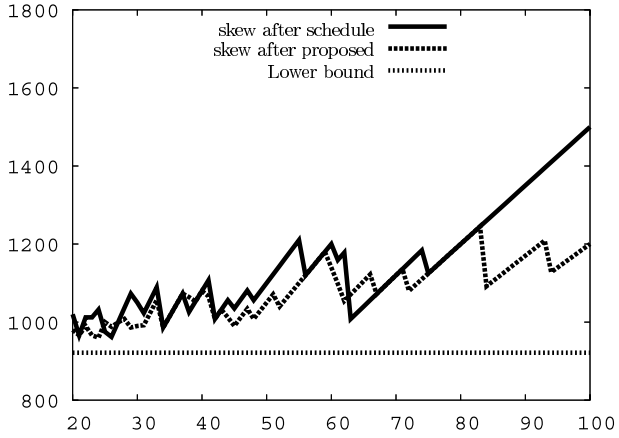

Fig. 15 Application time ($CS \times clk$) vs. clk for Lattice.

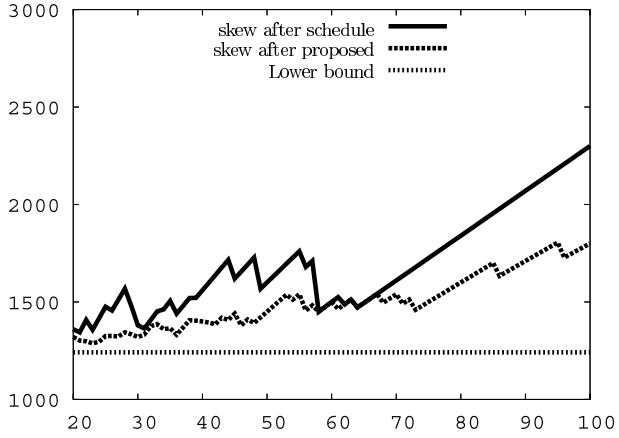

Fig. 16 Application time ($CS \times clk$) vs. clk for Elliptic.

ements. Note that area is presented with being normalized by the area of 2-input NAND gate (it has the area 1).

Considering the maximum delay of an adder, we assume that the clock period is set to 7 ns. Then we need to prepare delay elements up to 7 ns. In case that the skew for each register or multiplexer is controlled by its own delay element, the area overhead for each register or multiplexer

Table 2 Datapath components.

	16 bit adder	16 bit multiplier	16 bit register	16 bit reg. with reset	16 bit 2-1 MUX
Max. delay [ns]	5.82	17.79			
M in. delay [ns]	0.59	0.32			
Area	195	4329	176	209	64
Power [μW]	48.6501	2586	2.9154	7.4032	1.139

Table 3 Delay element.

Rising Delay [ns]	Falling Delay [ns]	Area	Power [uW]
0.38	0.43	4	0.2225
0.67	0.99	3	0.2225
1.45	1.48	5	0.5975
1.79	1.90	4	0.5700
2.18	2.45	8	0.9450
2.94	2.95	12	1.5562
3.49	3.49	6	0.9175
3.69	3.90	13	1.6675
4.44	4.46	14	1.9662
4.97	5.00	15	2.1888
5.38	5.46	15	2.1888
5.92	5.95	19	2.6888
6.36	6.45	13	1.9875
6.89	6.92	20	2.9113
7.46	7.49	25	3.2375
7.94	7.95	19	2.8350
8.41	8.49	28	3.7587
8.94	8.96	32	4.1962
9.44	9.48	28	4.1062
9.92	9.95	33	4.4812
10.40	10.48	36	4.6062

varies from 0 to around 20, and the extra power varies from 0 to around 3 μW. If we assume that skew values to registers and multiplexers spread equally from 0 to 7 ns, the average area overhead per register or multiplexer is around 10 which is around 6% of an 16 bit register and around 16% of an 16 bit 2-to-1 multiplexer. On the other hand, the average extra power per register or multiplexer is around 1.5 μW which is around 50% of an 16 bit register and around 130% of an 16 bit 2-to-1 multiplexer. In case that various clock skew values are generated from a single delay chain, the area overhead and extra power may possibly be decreased.

7. Conclusion

We have introduced a novel optimization problem, simultaneous schedule (control step assignment) and skew optimization problem. We presented a proof of NP-hardness and a heuristic algorithm for the simultaneous control step and skew optimization under given clock period. The algorithm has the potential to play a central role in various scenarios of skew-aware RT level synthesis. A study of the relation between the simultaneous optimization of skew and

re-timing in logic level and our problem in RT level is one of the interesting future works.

Acknowledgement

The authors would like to thank anonymous reviewers for their comments and suggestions which improved the presentation of this paper. The authors also would like to thank Dr. Iwagaki, JAIST, for his helpful supports on experiments. 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.P. Fishburn, "Clock skew optimization," IEEE Trans. Comput., vol.39, no.7, pp.945–951, 1990.
[2] R.B. Deokar and S.S. Sapatnekar, "A graphtheoretic approach to clock skew optimization," Proc. IEEE Int. Symp. Circuits and Systems, pp.1.407–1.410, 1994.
[3] B.A. Rosdi and A. Takahashi, "An algorithm to calculate the minimum clock period of a semi-synchronous circuit that contains multi-clock cycle path," IEICE Technical Report, VLD2005-8, Aug. 2005.
[4] X. Liu, M.C. Papaefthymiou, and E.G. Friedman, "Retiming and clock scheduling for digital circuit optimization," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.21, no.2, pp.184–203, 2002.
[5] E. Kamibayashi, Y. Kohira, and A. Takahashi "Circuit modification method of semi-synchronous circuit," IEICE Technical Report, VLD2004-146, ICD2004-242, March 2005.
[6] Y. Kohira and A. Takahashi, "Clock period minimization method of semi-synchronous circuits by delay insertion," IEICE Trans. Fundamentals, vol.E88-A, no.4, pp.892–898, April 2005.
[7] S.-H. Huang, C.-H. Cheng, Y.-T. Nieh, and W.-C. Yu, "Register binding for clock period minimization," Proc. 43rd Annual Conference on Design Automation, pp.439–444, July 2006.
[8] T. Obata and M. Kaneko "Simultaneous schedule and skew assignment for multiplexer control in placed datapaths," IEICE Technical Report, CAS2004-74, Jan. 2005.
[9] T. Obata and M. Kaneko, "Control signal skew scheduling for RT level datapaths," Proc. IEICE 18th Karuizawa Workshop on Circuits and Systems, pp.521–526, April 2005.
[10] T. Obata and M. Kaneko, "Control signal skew scheduling in RT level datapath synthesis," Proc. IEEE International Midwest Symposium on Circuits and Systems, CD–ROM ISBN:0-7803-9198-5, Aug. 2005.
[11] T. Obata and M. Kaneko, "Simultaneous control-step and skew assignment for control signals in RT-level datapath synthesis," Proc. SASIMI2006, pp.314–321, April 2006.
[12] J.P. Weng and A.C. Parker, "3D scheduling: High-level synthesis with floorplanning," Proc. Design Automation Conf., pp.668–673, 1991.
[13] Y.M. Fang and D.F. Wong, "Simultaneous functional unit binding and floorplanning," Proc. Int. Conf. on Computer Aided Design, pp.317–321, 1994.
[14] V.G. Moshnyaga and K. Tamaru, "A placement driven methodology for high-level synthesis of sub-micron ASIC's," Proc. Int. Symp. on Circuits and Systems, vol.4, pp.572–575, 1996.
[15] S. Tarafdar, M. Leeser, and Z. Yin, "Integrating floorplanning in data-transfer based high-level synthesis," Proc. Int. Conf. on Computer Aided Design, pp.412–417, 1998.
[16] P. Prabhakaran and P. Banerjee, "Parallel algorithm for simultaneous scheduling, binding and floorplanning in high-level synthesis," Proc. Int. Symp. on Circuits and Systems, vol.6, pp.372–376, 1998.
[17] K. Ohashi, M. Kaneko, and S. Tayu, "Assignment-space exploration approach to concurrent datapath/floorplan synthesis," Proc. Int. Conf. on Computer Design, pp.370–375, 2000.
[18] D. Kim, J. Jung, S. Lee, J. Jeon, and K. Choi, "Behavior-to-placed RTL synthesis with performance-driven placement," Proc. Int. Conf. on Computer Aided Design, 2001.
[19] M. Kaneko and K. Ohashi, "Assignment constrained scheduling under max/min logic/interconnect delays for placed datapath," Proc. APCCAS2004, vol.2, pp.545–548, 2004.

Appendix: Proof of NP-Hardness of (σ, τ)-Optimization Problem

We show the detailed proof of NP-hardness of (σ, τ)-Optimization Problem (Theorem 2).

Our proof is based on the polynomial time reduction from 3SAT to the decision version of (σ, τ)-optimization problem (In the following, we call it (σ, τ)-decision problem in short.).

First, we define the transformation from an instance of 3SAT problem to an instance of the (σ, τ)-decision problem. Let (X, C) be an instance of 3SAT problem, where $X = \{x_1, x_2, \cdots, x_n\}$ is a set of variables, $C = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is a formula, and for each clause $c_i = (l_{i1} \vee l_{i2} \vee l_{i3})$, literal l_{ij} is either x_k or $\neg x_k$ for some k. An input instance of the decision version of (σ, τ)-optimization problem is a 8-tuple $(G, \rho, \xi, next, D, d, clk, CS_{spec})$, and the problem asks "Are there any feasible σ and τ satisfying that the maximum control step is less than or equal to CS_{spec}?"

Data Flow Graph $G = (\mathcal{O}, \mathcal{D})$:

The set of vertices $\mathcal{O}$ is;

$$\mathcal{O} = \{O_{c_i} | \, 0 \le i \le m\} \cup \{O_{l_{ij}} | \, 1 \le i \le m, 1 \le j \le 3\},$$

and the set of edges $\mathcal{D}$ is;

$$\mathcal{D} = \{(O_{c_{i-1}}, O_{c_i}) | 1 \le i \le m\} \cup \{(O_{c_{i-1}}, O_{l_{i1}}), (O_{l_{i1}}, O_{l_{i2}}), (O_{l_{i2}}, O_{l_{i3}}), (O_{l_{i3}}, O_{c_i}) | 1 \le i \le m\}$$

Please refer to Fig. A· 1.

Resource assignments ρ and ξ:

For operations, they are assigned to separate FUs (no FU sharing occurs). For data, we prepare $n + 1$ registers named $x_1, x_2, \cdots, x_n$, and y, and we assign $\xi(O_{c_0}) = \xi(O_{c_1}) = \cdots = \xi(O_{c_m}) = y$, and $\xi(O_{l_{ij}}) = x_k$, where the literal l_{ij} is either x_k or $\neg x_k$, for all i and j.

next:

Trivial from the data dependency specified by $G = (\mathcal{O}, \mathcal{D})$.

Maximum path delay D and minimum path delay d:

Maximum path delays are set depending on the number

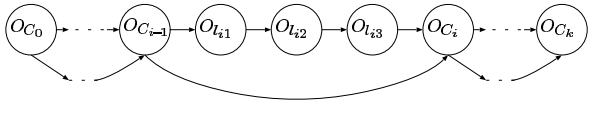

Fig. A· 1 A DFG for a 3SAT instance.

of negated variables in a clause.

1. For a clause $c_i = (x_j \vee x_k \vee x_l)$; $D^{O_{li1}}_{y-x_j} = D^{O_{li2}}_{x_j-x_k} = D^{O_{li3}}_{x_k-x_l} = D^{O_{ci}}_{x_l-y} = 0.5$, $D^{O_{ci}}_{y-y} = 3$.
2. For a clause $c_i = (x_j \vee x_k \vee \neg x_l)$; $D^{O_{li1}}_{y-x_j} = D^{O_{li2}}_{x_j-x_k} = 0.5$, $D^{O_{li3}}_{x_k-x_l} = D^{O_{ci}}_{x_l-y} = 1$, $D^{O_{ci}}_{y-y} = 4$.
3. For a clause $c_i = (x_j \vee \neg x_k \vee \neg x_l)$; $D^{O_{li1}}_{y_1-x_j} = D^{O_{li3}}_{x_k-x_l} = 0.5$, $D^{O_{li2}}_{x_j-x_k} = D^{O_{ci}}_{x_l-y} = 1$, $D^{O_{ci}}_{y-y} = 4$.
4. For a clause $c_i = (\neg x_j \vee \neg x_k \vee \neg x_l)$; $D^{O_{li1}}_{y-x_j} = D^{O_{ci}}_{x_l-y} = 1$, $D^{O_{li3}}_{x_k-x_l} = D^{O_{li2}}_{x_j-x_k} = 0.5$, $D^{O_{ci}}_{y-y} = 4$.

On the other hand, we can assign an arbitrary value (between 0 and the corresponding maximum path delay) to the minimum path delay d^o_{x-y}, i.e., $0 < d^o_{x-y} \le D^o_{x-y}$.

clk:

We set $clk = 1$.

CS_{spec}:

We set $CS_{spec} = 3m_0 + 4(m - m_0)$, where m is the total number of clauses and m_0 is the number of clauses including only non-negated variables.

Others:

We set s, h, t_{err} to 0, for the simplicity purpose. Discussions do not lose the generality by this simplification, because we can generate an equivalent problem instance with non-zero s, h, t_{err} by simply arranging the maximum path delay $D^o_{x-x'}$ to $D^o_{x-x'} - t_{err} - s$ and the minimum path delay $d^o_{x-x'}$ to $d^o_{x-x'} + t_{err} + h$. Based on this observation, the following discussions are made under $s = h = t_{err} = 0$.

The basic idea behind this transformation is to simulate the $\{0, 1\}$ assignment to variables in 3SAT problem by the skew assignment to registers $x_1, x_2, \cdots, x_n$ in the (σ,τ)-decision problem.

The first lemma ensures that the optimum solution for a problem instance obtained by the above transformation from an instance of 3SAT problem can be determined without depending on the minimum path delay information, and we need to care only maximum path delays.

Lemma 1. *Let o and o' be distinct operations and there exist an edge (o', o) in G. If there exist a directed path from o to $next(\xi(o'), o')$ in G, the minimum path delay $d^o_{\xi(o')-\xi(o)} \ge 0$ does not affect $\sigma(c^o_{\xi(o)})$.*

Proof. If there exist a directed path from o' to $next(\xi(o'), o')$ in G, we can sum up the setup constraints of registers along this path, and we have

$$\sigma(c^o_{\xi(o)}) + \tau(\xi(o)) \le \sigma(c^{next(o',\xi(o'))}_{\xi(o')}) + \tau(\xi(o')).$$

It means that the hold constraint

$$\sigma(c^o_{\xi(o)}) \cdot clk + \tau(\xi(o)) \le \sigma(o^{next(o',\xi(o'))}_{\xi(o')}) \cdot clk + \tau(\xi(o')) + d^o_{\xi(o')-\xi(o)}$$

is satisfied automatically without depending on the value of $d^o_{\xi(o')-\xi(o)}$. □

Next, we will introduce the following lemma which allows us to restrict the skew value to doubleton $\{0, 0.5clk\}$. By this restriction, we can ensure the correspondence between $\{0, 1\}$ assignment in 3SAT and the $\{0, 0.5clk\}$-skew assignment in the (σ,τ)-decision problem.

Lemma 2. *If the clock period is clk and each of maximum and minimum path delays has the value either $k \cdot clk$ or $(k + 0.5)clk$ for some integer k, there always exists a (σ, τ)-optimum solution with only 0, $0.5clk$ skew values. (From an arbitrary (σ, τ)-optimum solution, we can construct a (σ, τ)-optimum solution having only 0, $0.5clk$ skew values. The time complexity of this transformation is $O(|\mathcal{M}|)$.)*

Proof. Let σ and τ be an optimum solution of the (σ, τ)-optimization for an input instance in which every path delay has the form either $k \cdot clk$ or $(k + 0.5)clk$ for some integer k. Let τ^* be the transformed version of τ, which is obtained as follows.

$$\tau^*(x) = \begin{cases} 0 & \text{if } 0 \le \tau(x) < 0.5clk \\ 0.5clk & \text{if } 0.5clk \le \tau(x) < clk \end{cases} \quad (\text{A}\cdot 1)$$

Note that $\tau(x) \ge \tau^*(x)$ for all x. It will be proven that σ and τ^* is also an optimum solution for the given instance. Since σ is unchanged, it is enough to verify that the pair of σ and τ^* is a feasible solution. For the setup constraint of any part

$$\sigma(c^o_x) \cdot clk + \tau(x) \ge \sigma(c^{o'}_{x'}) \cdot clk + \tau(x') + D^o_{x-x'},$$

we have

$$\sigma(c^o_x) \cdot clk + \tau^*(x) + (\tau(x) - \tau^*(x)) - (\tau(x') - \tau^*(x')) \ge \sigma(c^{o'}_{x'}) \cdot clk + \tau^*(x') + D^o_{x-x'}$$

Since $\sigma(c^o_x) \cdot clk$, $\tau^*(x)$, $\sigma(c^{o'}_{x'}) \cdot clk$, $\tau^*(x')$ and $D^o_{x-x'}$ are all multiples of $0.5clk$, and $-0.5clk < (\tau(x) - \tau^*(x)) - (\tau(x') - \tau^*(x')) < 0.5clk$, we can truncate $(\tau(x) - \tau^*(x)) - (\tau(x') - \tau^*(x'))$ to 0, and we have

$$\sigma(c^o_x) \cdot clk + \tau^*(x) \ge \sigma(c^{o'}_{x'}) \cdot clk + \tau^*(x') + D^o_{x-x'}$$

We can verify hold constraints for σ and τ^* in a similar way, and we can conclude the feasibility of the pair σ and τ^*. □

From Lemmas 2 and 1, we can discuss an optimum solution for the problem instance obtained by the transformation from 3SAT only considering maximum path delays and doubleton $\{0, 0.5\}$ for skew values.

Proof of Theorem 1: It is clear that the (σ, τ)-decision problem is in the class NP.

The transformation from an instance of 3SAT $((X, C))$ to an instance of the (σ, τ)-decision problem $((G, \rho, \xi, next, D, d, clk, CS_{spec}))$ can be done in polynomial time. We claim that $(G, \rho, \xi, next, D, d, clk)$ has σ and τ such that $\sigma(c^{O_{cm}}_y) - \sigma(c^{O_{c0}}_y)$ is less than or equal to $3m_o + 4(m - m_0)$ if and only if (X, C) is satisfiable. It can be easily verified that $\sigma(c^{O_{cm}}_y) - \sigma(c^{O_{c0}}_y)$ equals to $3m_o + 4(m - m_0)$ if and only if

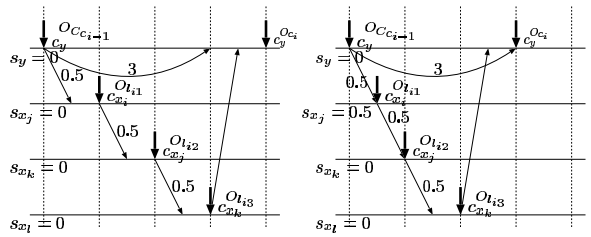

Fig. A· 2 2 control-step assignments for the clause $c_i = (x_j \vee x_k \vee x_l)$.

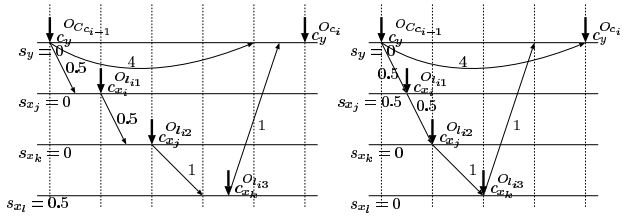

Fig. A· 3 2 control-step assignments for the clause $c_i = (x_j \vee x_k \vee \neg x_l)$.

$$\sigma(c_y^{O_{c_i}}) - \sigma(c_y^{O_{c_{i-1}}}) = \begin{cases} 3 & \text{if } c_i \text{ includes non-negated variables only,} \\ 4 & \text{if } c_i \text{ includes at least one negated variable} \end{cases}$$

is satisfied for all i, $1 \le i \le m$. For a clause $c_i = (x_j \vee x_k \vee x_l)$ (all non-negated variables), $\sigma(c_y^{O_{c_i}}) - \sigma(c_y^{O_{c_{i-1}}}) = 3$ if and only if at least one from $\tau(x_j)$, $\tau(x_k)$ and $\tau(x_l)$ equals to 0.5 (See Fig. A· 2), which corresponds to the fact that at least one from x_j, x_k and x_l is assigned 1 and the clause c_i is satisfied. The similar arguments are satisfied for the other types of clauses (Fig. A· 3 shows one other case where the clause includes exactly one negated variable). So, we can conclude that $\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}})$ equals to $3m_o + 4(m - m_0)$ if and only if all clauses are satisfied. □

Takayuki Obata received his B.E. degree in electrical and electronics engineering from Tokyo Institute of Technology, Tokyo, Japan, in 2001, and M.S. degree in information science from Japan Advanced Institute of Science and Technology, Ishikawa, Japan, in 2003. He is currently working toward his Ph.D. degree at Japan Advanced Institute of Science and Technology. His main research interest is high-level synthesis.

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