Advanced Datapath Synthesis Incorporating Intentional Timing Skew for High Performance Nanometer VLSIs

by

Takayuki OBATA

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Mineo Kaneko

School of Information Science
Japan Advanced Institute of Science and Technology

March 2009

Abstract

In the logic level VLSI design, the clock skew is now utilized intentionally for improving system performances and significant efforts have been devoted. Similar to the clock schedule in the logic level design, the skew-aware high level design will contribute to reducing the clock period. In addition, the intentional skew considered in high level synthesis may also contribute to reducing the number of control steps (makespan) for a target application. In the logic level design, the effect of the intentional clock skew is often enhanced by re-timing technique. 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.

In this thesis, we investigate the optimization of schedule (σ), skew (τ) and clock period (clk). We assume that resource binding and delay information are given as an a part of input description. The contributions of this thesis are following.

- The proof of NP-hardness of simultaneous optimization of (σ,τ,clk).
- The proof of NP-hardness of simultaneous optimization of (σ,τ) under given clk.
- The proof of NP-completeness of decision problem whether there exists a feasible pair of (σ,τ) for the input instance under given clk.
- A sufficient and necessary condition for the input instance to have a feasible pair of (σ,τ) for any clk. This condition is also a sufficient condition to have a feasible pair of (σ,τ) for specified clk.
- A heuristic algorithm for simultaneous optimization of (σ,τ) under given clk. The objective of the algorithm is to minimize the number of control steps.

Intentional skew control is a promising key technology not only to improve VLSI performance, but also to provide tunability for each VLSI to operate with its own maximum performance, which may overcomes the current and future process variability problem. Those results presented in this paper should be important theoretical base of skew-aware datapath design.

Acknowledgments

The author wishes to express his sincere gratitude to his principal advisor Professor Mineo Kaneko of Japan Advenced Institute of Science and Techology for his constant encouragement and kind guidance during this work.

The author also wishes to express his thanks to Professor Yasushi Hibino, Kunihiko Hiraishi and Atsuko Miyaji of Japan Advanced Institute of Science and Technology and Associate Professor Atsushi Takahashi of Tokyo Institute of Technology for their suggestions and continuous encouragements.

The author is grateful to all who have affected or suggested his areas of research.

The author devotes his sincere thanks and appreciation to all of them, and his colleagues.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

High level synthesis is the transformation of an input algorithm which is required to be implemented on VLSI to a RT-Level architecture. A RT-Level architecture consists of components such as registers, multiplexers, and FUs and connections between components. An input algorithm consists of operations and data dependencies between operations. To generate RT-Level architecture, we have to assign operations and relevant data to FUs and registers. This assignment is called "resource binding". A register reads an output of an operation and a multiplexer selects a data for a FU to execute an operation or for a register to read an output. These actions are controlled by control signals which are assigned to discrete time slots called "control steps" and are issued synchronized with a clock signal. We call this assignment of control signals to control steps as "control schedule."

In deep sub-micrometer or nanometer technology, interconnection delay becomes a dominant factor for the operation speed of VLSI systems. To synthesize high performance VLSI systems, the importance of exploiting interconnection delay information at higher level design is recognized, and several synthesis systems, which combine tasks in high level synthesis and floorplan, have been proposed [1]-[5]. When we determine the control schedule directly considering not only maximum delays but also minimum delays, we can temporally overlap some data flows in same data path to improve system performance. We can see one typical example of it in [8].

To improve system performance in terms of total computation time and/or robustness to delay variation, we are going to introduce appropriate delays which differentiate the arrival times of control signals to registers and multiplexers. A similar technique to the skew of register control has been proposed for sequential circuits, and significant efforts have been devoted to so–called clock scheduling [9],[10],[11],[12]. 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 [13],[14]. 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 and temporal order of lifetimes.

In this thesis, we investigate the optimization of schedule (σ), skew (τ) and clock period (clk). In Chapter 2, we show some examples which show the effectiveness of simultaneous schedule, skew and clock period optimization and formulate the simultaneous optimization problem. We also present an approximation algorithm for skew and clock period optimization in this chapter. It is based on graph algorithm and binary search. In Chapter 3, we give a proof of NP-hardness of simultaneous schedule, skew and clock period optimization problem by reduction from 3SAT problem. Following Chapters are about partial problems. In Chapter 4, we show a proof of NP-hardness of simultaneous skew and schedule optimization and NP-completeness of decision problem whether the input has a feasible solution or not. In Chapter 5, we present a sufficient condition for the decision problem whether simultaneous skew and schedule assignment problem have a feasible solution or not. We present a heuristic algorithm for simultaneous skew and schedule optimization problem in Chapter 6.

Chapter 2

Preliminaries

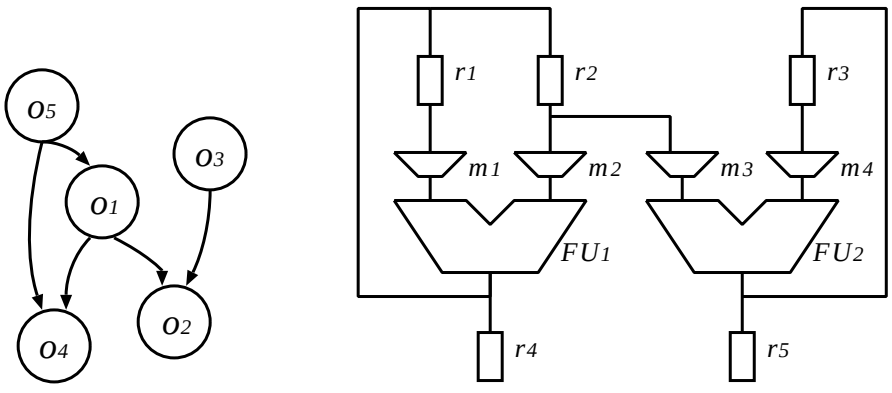
2.1 Background and Motivational Example

2.1.1 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 binding, 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 the set of functional units, $\mathcal{R}$ is the set of registers, $\mathcal{U}$ is the set of operations whose outputs are not written to registers, and $\xi(o) = r$ means that the output data of operation o is assigned to 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 a datapath circuit obtained from Fig.2.1(a) with the resource binding $\rho(o1) = FU1, \rho(o2) = FU2, \rho(o3) = FU2, \rho(o4) = FU1, \rho(o5) = FU1, \xi(o1) = r2, \xi(o2) = r5, \xi(o3) = r3, \xi(o4) = r4, \xi(o5) = r1$.

The behavior of a datapath circuit, on the other hand, is determined by the arrival

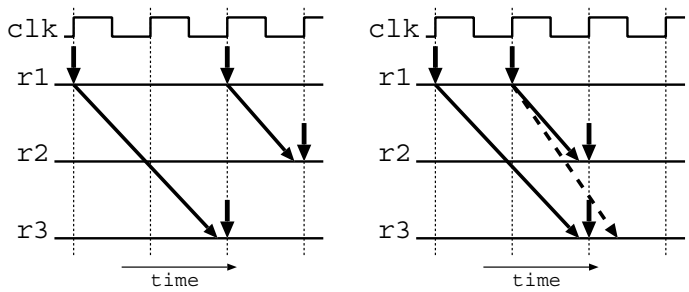


(a) (b)

Figure 2.1: Example of DFG and RT-Level architecture.

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 $x \in \mathcal{M}$. Each $c_x^o \in \mathcal{S}$ will be assigned to an appropriate control step. The control step is denoted as $\sigma(c_x^o)$ and we call control step assignment $\sigma : \mathcal{S} \to \mathbb{Z}_+$ as a control schedule. Timing skew assignment $\tau : \mathcal{M} \to \mathbb{R}$ assign skew between arrival timing of a control signal and begging of a control step. 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.

2.1.2 Motivational Example



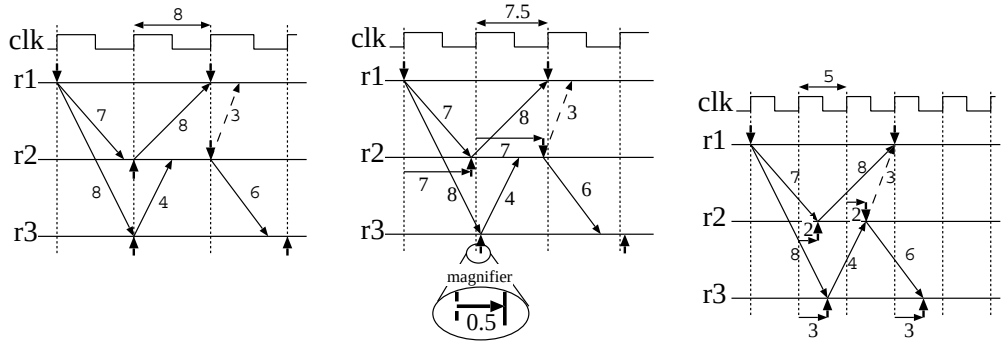
(a) A schedule with only maximum delay

(b) A schedule with maximum and minimum delay

Figure 2.2: Benefit of considering minimum delay

In conventional High Level Synthesis, a schedule is decided based on maximum delay. In this paper, we discuss scheduling considering both maximum and minimum delays of a signal transmission path which includes a functional unit. Fig.2.2 shows the benefit of considering minimum delays. Each horizontal line indicates the time axis at a multiplexer or a register. A vertical arrow shows the arrival of a control signal and a solid slant arrow shows a data propagation. A broken slant arrow shows unintended data propagation which may break a necessary data. A data propagation from r1 to r3 starts when the first control signal arrives at r1. The unintended data propagation from r1 to r3 starts when the second control signal arrive at r1. The schedule based on only maximum delays (Fig.2.2(a)) needs 4 control steps, while the schedule considering maximum and minimum delays(Fig.2.2(b)) needs 3 control steps. The start of data propagation is controlled by control signals for multiplexers and registers. This is the reason to schedule control signals directly.

Fig.2.3(a) shows a schedule of 6 control signals. The number written beside a slant solid/broken arrow shows maximum/minimum delay of the data propagation. The schedule requires 4 control steps. This is an optimal schedule under zero skew if the number of control steps is restricted to smaller than or equal to 4, and its minimum clock period is 8 (total computation time $8 \times 4 = 32$). When we assign skew $(\tau_{r1}, \tau_{r2}, \tau_{r3}) = (0, -0.5, 0.5)$, the minimum clock can be reduced to 7.5 (total computation time $7.5 \times 4 = 30$). The situation is illustrated in Fig.2.3(b). Note that the negative skew value assigned to r2 is transformed to 7 as described in Section 2.2.1. Fig.2.3(c) shows an optimal schedule and skew assignment. The minimum clock period is now 5 (totally, $5 \times 4 = 20$). This example



(a) An optimal schedule with zero skew

(b) Skew optimization for 2.3(a)

(c) An optimal schedule and skew assignment

Figure 2.3: Necessity of skew aware scheduling

shows that we can reduce clock period by optimizing skew value for existing zero skew schedule, but we may not generate optimal schedule in such a way, so we should decide schedule and skew simultaneously.

Skew optimization was discussed mainly for sequential logic circuits, where the subject to be optimized is the arrival time difference of the clock signal instead of control signals. In deep submicrometer or nanometer technology, interconnection delay becomes a dominant factor for the operation speed of a VLSI system, and the importance of exploiting interconnection delay information at higher level design is recognized. Coupled with this interconnection delay information, control signal schedule and skew assignment based on maximum/minimum delay model will be important techniques supporting the design of high performance VLSIs.

The main subject of this paper is the simultaneous skew and control schedule optimization in high-level datapath synthesis.

2.2 Simultaneous Optimization of Control Step and Skew Assignments

2.2.1 Formulation of the Problem

Our simultaneous optimization problem, (σ,τ,clk)-optimization, receives a data flow graph G, resource binding 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.

Fig.2.4 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.

The "setup constraint" (the arrival of the control signal $c_{r_j}^{o_j}$ has to be later than the

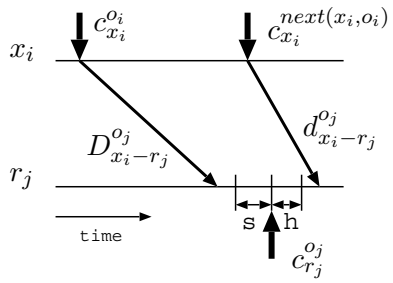


Figure 2.4: Setup / Hold constants.

arrival of the result of o_j) is formulated as

$$\sigma(c_{x_i}^{o_*}) \cdot clk + \tau(x_i) + t_{err} + D_{x_i-r_j}^{o_j} + s \leq \sigma(c_{r_j}^{o_j}) \cdot clk + \tau(r_j)$$

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. $D_{x_i-r_j}^{o_j}$ 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 $c_{r_j}^{o_j}$ has to be earlier than the destruction of the result of o_j) is given as

$$\sigma(c_{r_j}^{o_j}) \cdot clk + \tau(r_j) + t_{err} \leq \sigma(c_{x_i}^{next(x_i,o_i)}) \cdot clk + \tau(x_i) + d_{x_i-r_j}^{o_j} - h,$$

where $d_{x_i-r_j}^{o_j}$ 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_{m-r}^{o}s have almost the same value, and also the minimum path delays d_{m-r}^{o}s have almost the same value. So we can set $\sigma(c_m^o)$ and $\tau(m)$ as following.

$$\sigma(c_m^o) = \lfloor \frac{\sigma(c_r^o) \cdot clk + \tau(r) - D_{m-r}^{o}}{clk} \rfloor,$$

$$\tau(m) = \tau(r) - D_{m-r}^{o} - \lfloor \frac{\tau(r) - D_{m-r}^{o}}{clk} \rfloor clk.$$

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 $\sigma : S \to \mathbb{Z}_+$ into $\sigma : S \to \mathbb{R}_{\geq \not\prec}$. We will call $\sigma : S \to \mathbb{R}_{\geq \not\prec}$ 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 (2.2.1)-(2.2.1) assuming variables τ equal to zero.

$$\sigma(c_{r_j}^{o_j}) \geq \sigma(c_{x_i}^{o_i}) + D_{x_i - r_j}^{o_j} + s + t_{err} \tag{2.1}$$

$$\sigma(c_{x_i}^{next(x_i,o_i)}) \geq \sigma(c_{r_j}^{o_j}) - d_{x_i - r_j}^{o_j} + h + t_{err} \tag{2.2}$$

That is, corresponding to (2.1)-(2.2), 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 $clk \times CS \to min$) 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.2.5. 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.

Figure 2.5: A simultaneous optimization algorithm using MILP

2.2.2 Partial Problems

Because simultaneous optimization of σ, τ, clk is a hard problem, we consider partial problems. The optimization of one of σ, τ, clk is easy. When we try to optimize σ or τ while other is fixed, we can use the longest path algorithm on the constraint graph described in section 5.3.1 and 5.3.2. The computation of clk is the easiest problem among three problems. We only have to compute the common region of one variable inequalities. 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. This problem is the same as the clock skew optimization problem in logic level. Because τ and clk take real values, (τ, clk)-optimization problem can be formulated as LP problem. We present an approximation algorithm in Section 2.3.

(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. We propose An efficient graph theoretic approach in Chapter 6.

2.3 A Solution Algorithm for Skew Optimization for Clock Period Minimization

In this section, we show a constant approximation algorithm CSO (Control Skew Optimization) for (τ,clk)-optimization problem. Algorithm CSO output clock period $clk \leq clk_{opt} + \epsilon$ and skew assignment τ in time $O(|\mathcal{O}|^2 \cdot log_2((clk_{max} - clk_{min})/\epsilon))$ where clk_{opt}, ϵ, clk_{max} and clk_{min} are minimum clock period for given input instance, accuracy specified by user, maximum bound of clock period given by user and minimum bound computed from setup and hold constraints, respectively.

(τ,clk)-optimization Problem is a same problem as clock skew optimization problem in logic level. The contribution in this section is also the first solution algorithm for clock skew optimization for the circuit which have multi clock cycle path. More advanced method for clock skew optimization problem have been proposed in [11].

2.3.1 Skew Scheduling for Minimizing Clock Period

From (2.2.1)-(2.2.1), we have

$$\tau(r_j) - \tau(x_i) \geq (\sigma(c_{x_i}^{o_i}) - \sigma(c_{r_j}^{o_j})) \cdot clk + \tau_{err} + D_{x_i - r_j}^{o_j} + s, \tag{2.3}$$

$$\tau(x_i) - \tau(r_j) \geq (\sigma(c_{r_j}^{o_j}) - \sigma(c_{x_i}^{next(x_i, o_i)})) \cdot clk + \tau_{err} - d_{x_i - r_j}^{o_j} + h. \tag{2.4}$$

We generate skew constraint multigraph $G_s = (V, E)$ from (2.3-2.4) as shown in Fig.2.6. Vertices V is a set of multiplexers, registers and one auxiliary source node v_s. Weighted edges E are the union of edges corresponding to (2.3)-(2.4) over all operations, and $\{(v, v_s) | v \in V \setminus v_s\} \cup \{(v_s, v) | v \in V \setminus v_s\}$. Edge weights for $\{(v, v_s) | v \in V \setminus v_s\}$ and $\{(v_s, v) | v \in V \setminus v_s\}$ are $-clk$ and 0. Then, skew assignment (skew schedule) problem is

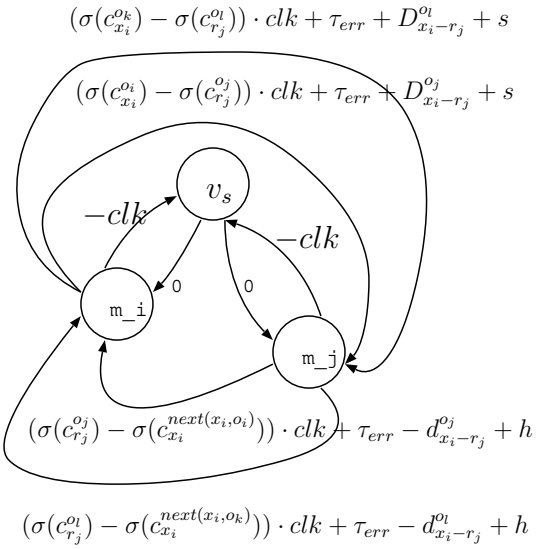


Figure 2.6: Skew constraint multigraph

now considered as the problem to assign real values to vertices in G_s, and maximum path lengths from v_s to other vertices gives us a solution, i.e., skew of registers and multiplexers. If G_s has a positive cycle, feasible skew schedule does not exist. Skew schedule for minimizing clock period is, then, the problem to find minimum clk such that the resultant G_s includes no positive cycle.

Once we assign a value to clk, multi edges between a pair of vertices can be reduced to one edge which has maximum weight (since only maximum path length is our concern), and hence we can use Ford algorithm to compute maximum path length.

The following lemma is a key to derive a binary search algorithm for finding minimum clock period.

Lemma 1 *There is at most one continuous clk region, for which feasible skew assignment exists.*

We assume that there is a positive cycle in G_s with $clk = clk^*, clk^* \in R_+$, the cycle weight can be denoted as $x \cdot clk^* + y$. We call x the "coefficient" of the cycle. If the coefficient x is positive, the cycle is also positive for all $clk > clk^*$. If the coefficient is negative, the cycle is also positive for all $clk < clk^*$. So we can derive the above lemma. (In conventional skew problem in a sequential circuit, $\sigma(c_{x_i}^{o_i}) - \sigma(c_{r_j}^{o_j}) = -1$ and $\sigma(c_{r_j}^{o_j}) - \sigma(c_{x_i}^{next(x_i,o_i)}) = 0$ hold equivalently, and all cycles have negative or zero coefficients.)

If upper and lower bound of clk is given, we can use binary search to find minimum clock period by identifying a positive cycle if it exists and extracting the polarity of its coefficient. Extraction of a cycle coefficient is performed using modified Ford algorithm whose complexity is the same with the original one.

The lower bound of clk "clk_{min}" is calculated from (2.3)-(2.4). clk_{min} must satisfy the following inequality so that $\tau(r_j) - \tau(x_i)$ has feasible range of value.

$$clk_{min} \geq \frac{D_{x_i-r_j}^{o_j} - d_{x_i-r_j}^{o_j} + s + h + 2\tau_{err}}{\sigma(c_{x_i}^{next(x_i,o_i)}) - \sigma(c_{x_i}^{o_i})} \tag{2.5}$$

The upper bound "clk_{max}" is assumed to be given by a user.

Step1. Generate G_s. Calculate clk_{min}.

Step2. If $clk_{max} - clk_{min} < \epsilon$, go to Step7.

Step3. Set $clk = (clk_{max} - clk_{min})/2$

Step4. Reduce multi edges of G_s. Apply modified Ford algorithm to compute longest path lengths on G_s.

Step5. If there is no positive cycle, set $clk_{max} = clk$ and go to Step2.

Step6. Extract a positive cycle. If its coefficient is positive, set $clk_{max} = clk$, If not, set $clk_{min} = clk$. Go to Step2.

Step7. Set $clk = clk_{max}$. Reduce multi edges of G_s and apply Ford algorithm. If there exist a positive cycle, given instance has no solution. If not, maximum path lengths to vertices are optimized control skews.

Figure 2.7: Algorithm CSO (Control Skew Optimization)

The algorithm CSO (Control Skew Optimization) is shown in Fig.2.7. "ϵ" is the resolution of clock period, and it is also given by a user. Fig.2.8 shows an illustrative example of our binary search done by CSO.

Theorem 2 *Algorithm CSO computes ϵ-minimum*[1] *clock period and feasible control skew for registers and multiplexers with $O(|\mathcal{O}|^2 \cdot log_2((clk_{max} - clk_{min})/\epsilon))$ computation time, if the length of the feasible control skew region is not smaller than ϵ.*

Computational complexity of CSO will be explained briefly. First, $|V| = O(|\mathcal{O}|)$ and $|E| = O(|\mathcal{O}|)$. The computational complexity for constructing G_s is $O(|\mathcal{O}|)$. Computation of clk_{min} and clk_{max} take $O(|E|)$ time. Let ϵ be time unit for clk. Binary search recurs

[1]There is no feasible solution which is smaller than ϵ-minimum solution minus ϵ.

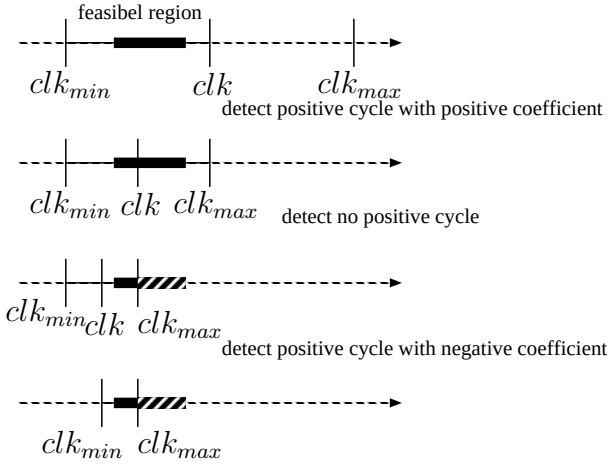


Figure 2.8: Example of binary search

$O(log_2((clk_{max} - clk_{min})/\epsilon))$ times. In each recurrence takes $O(|E|)$ time to remove multi edges and $O(V \cdot E)$ time for ford algorithm. Finally, proposed algorithm takes $O(|\mathcal{O}|^2 \cdot log_2((clk_{max} - clk_{min})/\epsilon))$ time.

Finally, it is commented that the algorithm CSO can be easily modified to find the maximum feasible clock period, if necessary. That is, we only need to modify CSO such that clk_{min} is updated if no positive cycle is detected in Step5, and set $clk = clk_{min}$ in Step7. Then we can compute the maximum feasible clock period in the same computational complexity with the original CSO.

2.3.2 Experiments

Proposed method was implemented using C programming language and tested on AMD Athlon based system. Our experimental instance are two DAG algorithms based on Jaumann wave filter, all-pole Lattice filter and Elliptic wave filter.

Delays between two modules are the sum of delays of register-multiplexer, multiplexer-FU, FU, FU-multiplexer, and multiplexer-register. Maximum/minimum delays of multipliers and adders are 60/10 and 20/10, respectively. The other delays are given randomly. minimum of register-multiplexer and FU-multiplexer delays are 3-25 and multiplexer-register and multiplexer-FU delays are 2-15. Maximum delay of a path is 1.1-1.4 times larger than its minimum delay.

clk reduction by skew

Table 2.1: Experiment 1

Jaumann

instance ID	#fu	#reg	n/s	w/s	ratio
1	9	10	47	31	0.66
2	9	9	51	32	0.62
3	9	10	51	41	0.80
4	9	11	51	36	0.70
5	9	10	66	52	0.79
6	9	12	68	60	0.88
7	9	12	69	59	0.86
8	10	11	70	59	0.84

Lattice

instance ID	#fu	#reg	n/s	w/s	ratio
1	6	8	64	40	0.63
2	6	8	42	27	0.64
3	6	8	56	38	0.68
4	6	8	66	44	0.67
5	6	8	56	40	0.71
6	6	8	64	57	0.89
7	6	8	59	56	0.95
8	5	8	72	58	0.81

In the first experiment, the effect of skew optimization is simply tested on several datapath designs. For each input instance (target algorithm), eight different control step schedule and binding are tested. Experimental results are shown in table 2.1. A user defined parameter clock resolution is 2. The column #fu and #reg represents the number of functional units and the number of registers in each datapath. The column n/s is the minimum clock period with zero skew. The column w/s is the minimum clock period with optimized skew obtained from our proposed algorithm. The column ratio is the reduction ratio of clock period. The results show the effectiveness of the skew optimization.

Table 2.2: Experiment 2

		totalcost without skew	*totalcost* with skew	*time* without skew	*time* with skew	CS	*controller*	*area*	*net*
Jaumann	competitor	1485.1	1349.1	812.55	676.50	7.0	107.8	26.2	44.4
	proposed	1693.1	1304.6	951.51	563.04	7.6	130.4	28.6	47.8
lattice	competitor	2113.4	1768.1	1521.3	1175.9	12.8	157.0	20.4	34.0
	proposed	2210.1	1752.9	1594.9	1137.7	11.8	156.0	20.6	36.8
elliptic	competitor	3880.7	3172.1	2402.1	1693.5	19.0	511.4	35.5	85.4
	proposed	3634.2	2816.6	2229.4	1411.8	17.4	458.4	34.7	83.6

Synthesis with skew optimization

The second experiment is the datapath synthesis enhanced by skew optimization. We combined proposed skew optimization with high-level synthesis system. The system is based on Simulated Annealing search of the solutions space which consists of assignment of operations to functional units, assignment of data to registers, order of operations on functional units, assignment of inputs of operations to port of functional units, and assignment of integers to operations (which indicate control step for the output of an operation to be written in an assigned register, not the timing of the execution of the operation). Control step for control signal to a multiplexer is optimized so that clock period without control skew is minimized.

We use *time*, *controller*, *net*, *area* to evaluate solutions. $time = CS \times clk$ where CS is the number of control steps. *controler* is the size of controler, which is proportional to the number of state and controlled modules. *net* is the number of connections between modules. *area* is the area of circuit. These four criteria are combined linearly to yield our objective function.

$$totalcost = \alpha \cdot time + \beta \cdot controller + \gamma \cdot net + \delta \cdot area$$

Coefficients α, β,γ, δ are decided experimentally as $\alpha = 1.0$, $\beta = 1.0$, $\gamma = 8.0$, $\delta = 8.0$.

We have carried out two types of synthesis, one is the synthesis with skew optimization and the other is the synthesis without skew optimization (all skews are fixed to 0). Results of both synthesis are evaluated with and without control skew optimization. Input algorithm is Jaumann wave filter, all-pole lattice filter and elliptic wave filter. Results are shown in table 2.2. Each value in the table is the average of 5 trials. As for parameters peculiar to simulated annealing, we use the following. Initial temperature is 1.0×10^5, terminate temperature is 0.50, temperature reduction ratio is 0.98, the number of iterations in inner loop is 14000 for competitor (zero skew version), 650 for proposed version (with skew optimization). Run time is 41m10s in average for proposed method, and 50m35s for competitor.

It seems natural that our proposed method can generate better solutions (datapaths) in both *totalcost* and *time* with skew, while these solutions are worse in *totalcost* and *time* without skew in jaumann and lattice cases. That is, the reversal in performance occurs, which indicates us the necessity of skew aware synthesis.

2.3.3 Conclusion

In this section, we have focused on the skew of the arrival time of control signals to multiplexers and registers in a RT level datapath. In interconnection delay dominant VLSI systems, the timing of the arrival of control signals at multiplexers should be carefully controlled as much as the timing of the arrival of control signals at registers.

An algorithm for this problem based on binary search has been developed, and it can be applied also to clock skew optimization of sequential circuits which have multi cycle paths. The computational time of proposed algorithm is about 0.7 times that of simplex algorithm when input algorithm is Jaumann wave filter or all-pole Lattice filter and 0.5 times when input algorithm is Elliptic wave filter. Proposed algorithm is more efficient for larger input instance.

Chapter 3

Computational Complexity of Simultaneous Optimization of Schedule, Skew and Clock Period

3.1 Introduction

In this chapter, we discuss the simultaneous optimization of the clock period (clk), the control step assignment (σ) and the skew assignment (τ) under given resource assignment and given operation order on each FU and Register. The contribution of this chapter is to show that our simultaneous optimization problem of schedule, skew assignment and clock period ((σ,τ,clk)-optimization problem in short) is HP-hard even if we limit the control step assignment to the one keeping the execution sequence of operations on a functional unit (and the lifetime sequence of data on a register) unchanged.

Although the contribution that (σ,τ,clk)-optimization problem is NP-hard do not give the method to solve the problem, it would important in the sense that it shows us a broad and crucial guideline for designing a solution algorithm. It implies that there is no polynomial time algorithm unless P=NP. It is a motivation to discussion about sub problems of (σ,τ,clk)-optimization problem in following chapters.

3.2 NP-hardness of (σ,τ,clk)-Optimization Problem

We show that (σ,τ,clk)-optimization problem is NP-hard by the reduction of MAX-3SAT problem to (σ,τ,clk)-optimization problem. Let (X, C) be the instance of MAX-3SAT problem, where $X = \{x_1, x_2, \cdots, x_n\}$ is the set of variables, $C = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is the algebra, and for each clause $c_i = (l_{i1} \vee l_{i2} \vee l_{i3})$, literal l_{ij} is either x_k or $\neg x_k$. The solution $\Xi : X \to \{0, 1\}$ for an instance (X, C) is the assignment of $\{0, 1\}$ for variables. We define a transformation from (X, C) to $(G, \rho, \xi, next, D, d, \lambda)$ and from (σ, s, clk) to Ξ.

We set (s,h,t_{err},λ) to (0,0,0,0.5).

We construct G as shown in fig.3.1. For each clauses c_i, we add operation O_{c_i}. O_{c_0} is auxiliary start operation. Between $O_{c_{i-1}}$ and O_{c_i}, there are two paths $O_{c_{i-1}}O_{l_{i1}}O_{c_i s_1}O_{l_{i2}}O_{c_i s_2}O_{l_{i3}}$ and $O_{c_{i-1}}O_{c_i w_1}O_{c_i w_2} \cdots O_{c_i w_{7-\alpha_i}}$ where α_i is the number of literal so as x_k. Operations $O_{l_{i1}}$, $O_{l_{i2}}$ and $O_{l_{i3}}$ correspond to literals l_{i1}, l_{i2} and l_{i3}, respectively.

We assign each operations to distinct FU. We tie in variable x_i with register x_i i.e. if

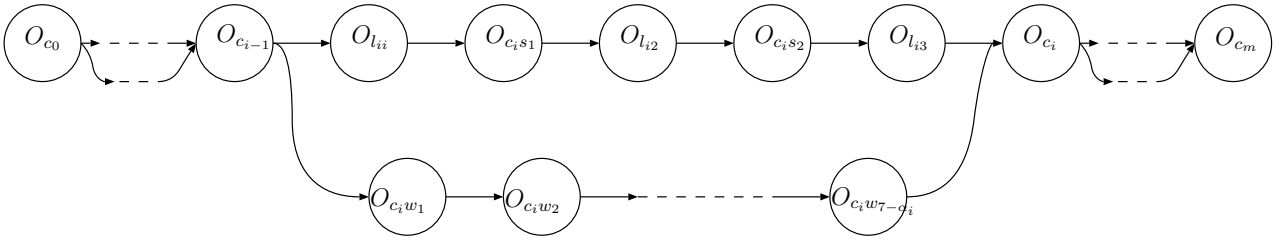


Figure 3.1: A DFG for a 3SAT instance.

$l_{ij} = x_k$ or $l_{ij} = \neg x_k$, $\xi(O_{l_{ij}}) = x_k$. To add to registers corresponding to variables, we use two registers y_1 and y_2. $\xi(O_{c_i}) = y_1$ and $\xi(O_{c_i w_j}) = y_2$. After we compute optimum solution (σ, τ, clk), we transform τ according to Lemma4, if $\tau(x_i) = \tau(y_1)$, we assign 0 for variable x_i, and if $\tau(x_i) \neq \tau(y_1)$, we assign 1 for variable x_i. Basic concept is to design $(G, \rho, \xi, n, D, d, \lambda)$ so that the more clauses are satisfied, the smaller CS is, and the number of satisfied clauses can be calculated from CS.

For each $O_{l_{ij}}$, let O_1, O_2 be the predecessor and the successor. As we show later, clk is 1 unit time in the optimum solution. Denote the minimum value of $\sigma_{y_1}^{O_2} - \sigma_{y_1}^{O_1}$ as L_{ij}. We assign delay value so that if $l_{ij} = 1$, L_{ij} is smaller than that with $l_{ij} = 0$. In case $l_{ij} = x_k$, we set $D_{y_1 - x_k}^{O_{l_{ij}}}, D_{x_k - y_1}^{O_2}$ to 0.5 unit time. If $\tau(x_k) \neq \tau(y_1)$(i.e. if $\tau(y_1) = 0$, $\tau(x_k) = 0.5$, and if $\tau(y_1) = 0.5$, $\tau(x_k) = 0$), L_{ij} is 1. On the other hand, if $\tau(x_k) = \tau(y_1)$, L_{ij} is 2. These situations are shown in fig.3.2. In case $l_{ij} = \neg x_k$, we set $D_{y_1 - x_k}^{O_{l_{ij}}} = D_{x_k - y_1}^{O_2} = 1$. If $\tau(x_k) = \tau(y_1)$, L_{ij} is 2. On the other hand, if $\tau(x_k) \neq \tau(y_1)$, L_{ij} is 3. These situations are shown in fig.3.3. Denote $\sigma_{y_1}^{O_{c_i}} - \sigma_{y_1}^{O_{c_{i-1}}}$ as χ_i. Then $\chi_i = 2\alpha_i + 3(3 - \alpha_i) = 9 - \alpha_i$ if c_i

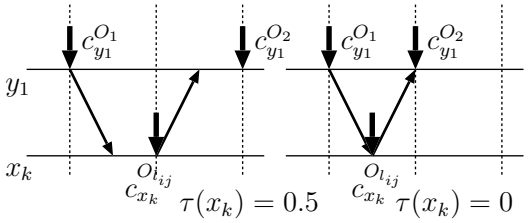


Figure 3.2: Edge weight for literal $l_{ij} = x_k$.

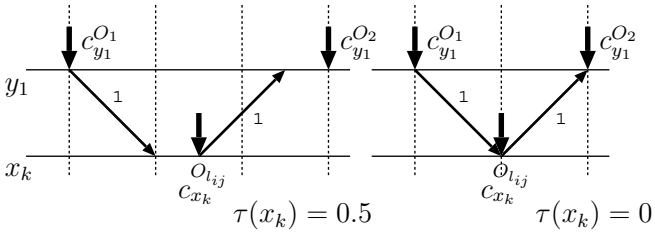


Figure 3.3: Edge weight for literal $l_{ij} = \neg x_k$.

is not satisfied, and $\chi_i < 9 - \alpha_i$ if c_i is satisfied. We use path $O_{c_{i-1}} O_{c_i w_1} O_{c_i w_2} \cdots O_{c_i w_{7-\alpha_i}}$ to assure $\chi_i > 8 - \alpha_i$. Maximum delay between registers (r_1, r_2), (r_2, r_2) and (r_2, r_1) for each operation in the path is 1.

Because $\chi_i = 9 - \alpha_i$ if c_i is not satisfied, or $\chi_i = 8 - \alpha_i$ if c_i is satisfied, so $CS = \sum_{i=1}^{m} \chi_i = 9m - \sum_{i=1}^{m} \alpha_i - z$ where z is the number of satisfied clauses, and thus if we minimize CS, the number of satisfied clauses is maximized.

We did not mentioned $next, d$. $next$ trivial for registers because there is a path between two operations O_1, O_2 such that $\xi(O_1) = \xi(O_2)$. We defined ρ so as to $\rho(O_1), \rho(O_2)$ for all $O_1 \neq O_2$. So we need not consider $next$ for multiplexers because there is no input multiplexer for any FU. We can assign arbitrary value to d^o_x in range $0 \leq d^o_{x-y} < D^o_{x-y}$ for any $x, y \in \mathcal{M}$ and $o \in G$ from following lemma.

Lemma 2 *Minimum delays do not affect σ and τ.*

Proof of Lemma: Let o, o' are operations such that $(o', o) \in G$. We prove that we can schedule $c^o_{\xi(o)}$ faster than $c^{next(o',\xi(o'))}_{\xi(o')}$; i.e. $\sigma^o_{\xi(o)} \cdot clk + \tau(\xi(o)) \leq \sigma^{next(o',\xi(o'))}_{\xi(o')} \cdot clk + \tau(\xi(o'))$. If $\sigma^o_{\xi(o)} \cdot clk + \tau(\xi(o)) \leq \sigma^{next(o',\xi(o'))}_{\xi(o')} \cdot clk + \tau(\xi(o'))$ hold constraint is obviously fulfilled for any $d^o_{\tau(o')-\tau(o)}$.

1. $o = O_{c_i w_1}$
   In this case, $\xi(o)$, o' and $\xi(o')$ is y_2, O_{c_i} and y_2, respectively. Because y_1 is the sole register whitch appear in left hand side of setup constraint (2.2.1) while y_2 is in right hand side, and $D^o_{y_1-y_2}$ is 1 for all o, we can set $\tau(y2)$ and $\sigma^{O_{c_i w_1}}_{y_2}$ as following without losing optimality.

$$\tau(y_2) = \tau(y_1) + 1 + \lfloor \frac{\tau(y_1) + 1}{clk} \rfloor clk,$$

$$\sigma^{O_{c_i w_1}}_{y_2} = \lfloor \frac{\sigma_{y_1} O_{c_i} + \tau(y_1) + 1}{clk} \rfloor$$

   Thus $\sigma^{O_{c_i w_1}}_{y_1} \cdot clk + \tau(y_2) = \sigma^{O_{c_i}}_{y_1} \cdot clk + \tau(y_1) + 1$. On the other hand, $next(O_{c_i}, y_1)$ is $O_{c_i s_1}$, and $\sigma^{O_{c_i s_1}}_{y_1} \cdot clk + \tau(y_1) \geq \sigma^{O_{c_i}}_{y_1} \cdot clk + \tau(y_1 + 1)$ Subjective inequality is fulfilled.

2. $o = O_{c_i w_j}, j \neq 1$
   In this case, $next(o', \xi(o')) = o$. It is obvious to fullfill the subjective inequality.

3. $o \neq O_{c_i w_j}$
   There exist a directed path from o' to $next(\xi(o'), o')$ in G, so tracing setup constraints of registers, we can verify $\sigma^o_{\xi(o)} \leq \sigma^{next(o',\xi(o'))}_{\xi(o')}$.

□

Finally, we show that $clk = 1$ and $\tau(x) \in \{0, 0.5\}$ in optimum solution.

Lemma 3 *$clk = 1$ in optimum solution.*

Proof of Lemma : We show that for each case of $clk \neq 1$, it is not a optimum solution if we set λ to 0.5. When we fix clk, let opt_{clk} be the value of optimum solution with given clk. CS_{clk} and $\chi_{i\,clk}$ denote the number of control steps and $\sigma^{O_{c_i}}_{y_1} - \sigma^{O_{c_{i-1}}}_{y_1}$ in opt_{clk}. From the constraint given by edges $(O_{c_i - 1}, O_{c_i w_1})$, $(O_{c_i w_j}, O_{c_i w_j + 1}) : 1 \leq j \leq 7 - \alpha_i - 1$ and $(O_{c_i w_{7-\alpha_i}}, O_{c_i})$, we have following inequality as we show in Fig.3.4.

$$\lceil 2/clk \rceil + \lceil 1/clk \rceil (7 - \alpha_i - 1) \leq \chi_{i\,clk}. \tag{3.1}$$

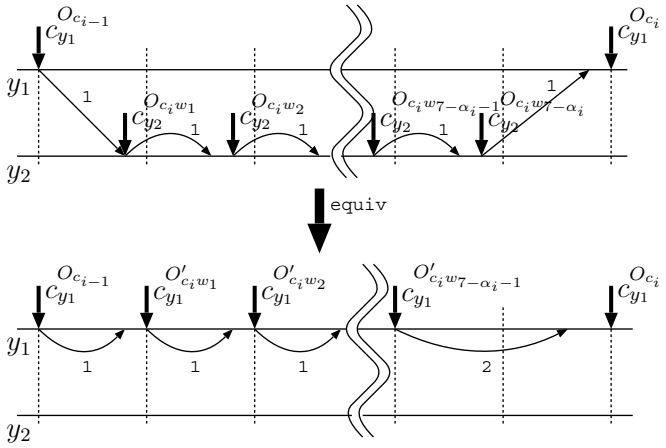


Figure 3.4: Two delay charts give equivalent two constraints for $\chi_{i\,clk}$ when we ignore the other path.

Because $CS_{clk} = \sum_{i=1}^{m} \chi_{i\,clk}$ and $3 \leq 7 - \alpha_i - 1 \leq 6$, we have

$$m\,(\lceil 2/clk \rceil + 3\lceil 1/clk \rceil) \leq CS_{clk}, \tag{3.2}$$

especially, $5m \leq CS_1$.

1. $clk < 0.5$
   There is no optimum solution in $clk < 0.5$ because $0.5CS_{0.5} = min_{clk} clk\, CS_{clk}$ and $CS_{clk} > CS_{0.5}$ in $clk < 0.5$.

2. $clk = 0.5$
   From (3.1), we have $\chi_{i\,0.5} \geq 16 - 2\alpha_i$ and $\chi_{i\,1} \leq 9 - \alpha_i$, so $\chi_{i\,0.5} \geq 2\chi_{i\,1} - 2$. It leads $CS_{clk} \geq 2CS_1 - 2m$.

$$\begin{aligned} opt_{0.5} - opt_1 &= CS_{0.5}(0.5 + \lambda) - CS_1(1 + \lambda) \\ &\geq 2CS_1 - 2m - 1.5CS_1 \\ &= 0.5CS_1 - 2m \\ &\geq 0.5(5m) - 2m = 0.5m > 0 \end{aligned}$$

3. $0.5 < clk < 2/3$
   In this case, $L_{kl} \geq 2$ for $l_{kl} = x_j$ from $clk < 1$, and $L_{kl} \geq 4$ for $l_{kl} = \neg x_j$ from $3clk < 2$. This fact means that we can reduce clk to 0.5 while CS is not changed. So we do not have optimum solution.

4. $2/3 \leq clk < 1$
   From (3.1), we have $\chi_{i\,clk} \geq 2(7-\alpha_i)+3$ and $\chi_{i\,1} \leq 7-\alpha_i+2$, so $CS_{clk} \geq 2CS_1 - m$.

$$\begin{aligned} opt_{clk} - opt_1 &\geq (2CS_1 - m)(clk + \lambda) - CS_1(1 + \lambda) \\ &= (2clk - 0.5)CS_1 - clk\, m - 0.5m \\ &\geq (2clk - 0.5)5m - clk\, m - 0.5m \\ &= 9m\, clk - 3m \\ &> 9m(2/3) - 3m = m > 0 \end{aligned}$$

5. $1 < clk < 1.5$
   For $l_{kl} = x_j$, $L_{kl} = 1$ if $0.5 \leq tau(x_j) \leq clk - 0.5$ and $L_{kl} = 2$ in other case. For $l_{kl} = \neg x_j$, $L_{kl} = 2$ if $1 \leq tau(x_j)$ or $tau(x_j) \leq clk - 1$, and $L_{kl} = 3$ for other case. So we can reduce clk to 1 while we do not change CS by setting $tau(x_j)$ to 0.5 if $0.5 \leq tau(x_j) \leq clk - 0.5$, and 0 in other case,

6. $1.5 \leq clk < 2$
   From (3.1), we have $\chi_{i\,clk} \geq 8 - \alpha_i$, so $CS_{clk} \geq CS_1 - m$.

$$\begin{aligned} opt_{clk} - opt_1 &\geq (CS_1 - m)(1.5 + \lambda) - CS_1(1 + \lambda) \\ &= 0.5CS_1 - 2m \geq 2.5m - 2m = m > 0 \end{aligned}$$

7. $2 \leq clk$
   From (3.1), we have $\chi_{i\,clk} > 7 - \alpha_i$, so $CS_{clk} \geq CS_1 - 2m$.

$$\begin{aligned} opt_{clk} - opt_1 &\geq (CS_1 - 2m)(2 + \lambda) - CS_1(1 + \lambda) \\ &= CS_1 - 5m = 0 \end{aligned}$$

□

Lemma 4 *If clock period is clk and delay values are either $k \cdot clk$ or $(k+0.5)clk$ for some integer k, we can transform skew values of optimum solution to 0 ore $0.5clk$ keeping σ in time complexity of $O(|\mathcal{M}|)$.*

Proof of Lemma : Let Δ_x is $\tau(x)$(if $\tau(x) < 0.5clk$) or $\tau(x) - 0.5clk$(if $\tau(x) \geq 0.5clk$) for every register x. We show that for all register such that $\Delta_x = min_{y \in \mathcal{R}} \Delta_y$, we can reduce skew value by Δ_x.

Consider $X = \{x | \Delta_x = min_{y \in \mathcal{R}} \Delta_y\}$ and suppose we can not reduce $\tau(x)$ for some $x \in X$. Then there exist a register x', operations o, o' and constraint inequality $\sigma_x^o + \tau(x) \geq \sigma_{x'}^{o'} + \tau(x') + D_{x-x'}^o$ so that $\sigma_x^o + \tau(x) - (\sigma_{x'}^{o'} + \tau(x') + D_{x-x'}^o) < \Delta_x$. Because σ is integer and $D_{x-x'}^o$ is multiple of $0.5clk$, $\Delta_{x'} \leq \Delta_x$. This is a contraction.

From above discussion, we can reduce skew value by Δ_x, i.e. reduce to 0 or $0.5clk$, keeping σ. Obviously, the time complexity of this transformation is $O(|\mathcal{M}|)$. □

Now we go on to the most important theorem.

Theorem 3 *(σ,τ,clk)-optimization problem is NP-hard.*

Proof of Lemma : The transform from an instance of MAX-3SAT to an instance of (σ,τ,clk)-optimization problem, the mapping from τ to $\{0, 0.5\}$, and the transformation from a solution of (σ,τ,clk)-optimization problem (σ, τ, clk) to a solution MAX-3SAT problem Ξ are in class of polynomial time complexity. We designed a transformation so as to $clk = 1$, so the objective of (σ,τ,clk)-optimization problem is to minimize CS. When we try to minimize CS, $\sigma_y^{O_{c_m}} - \sigma_y^{O_{c_0}}$ is also minimized. Because the smaller $\sigma_y^{O_{c_m}} - \sigma_y^{O_{c_0}}$, the larger the number of satisfied clauses, so we can solve MAX-3SAT problem by solving (σ,τ,clk)-optimization problem. Thus, we can reduce MAX-3SAT problem into (σ,τ,clk)-optimization problem, so (σ,τ,clk)-optimization problem is NP-hard. □

3.3 Conclusion

We have introduced a novel optimization problem, simultaneous control schedule and skew optimization problem, and we have proven that the problem with fixed clock period is NP-hard. The proof is based on the reduction from 3SAT problem. The result of this paper would become an important base for various types of the intentional-skew-aware system optimization problem. Development of an efficient heuristic algorithm for the simultaneous optimization of control schedule and skew assignment is our future target.

Chapter 4

Computational Complexity of Simultaneous Optimization and Solvability of Simultaneous Optimization of Schedule and Skew assignment

4.1 Introduction

The simultaneous optimization of the control step assignment (σ) and the control skew assignment (τ) is a powerful technique in improving performance. So the computational complexity of (σ,τ)-optimization problem is an important subject for a investigation. We show a NP-hardness of simultaneous optimization of (σ,τ) in this chapter. The proof is based on a reduction from one of most common NP-hard problem, 3SAT problem. A 3SAT instance is transformed into an instance of decision version of (σ,τ)-optimization problem which asks whether there exists a control step assignment which achieve a goal of the number of control steps.

An unfortunate result that (σ,τ)-optimization problem attract a controversy surrounding the tractability of computation of a feasible solution of (σ,τ)-problem. Our second contribution in this chapter is that a decision problem of solvability of an instance of (σ,τ)-problem.

4.2 NP-Hardness of (σ,τ)-Optimization Problem

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

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

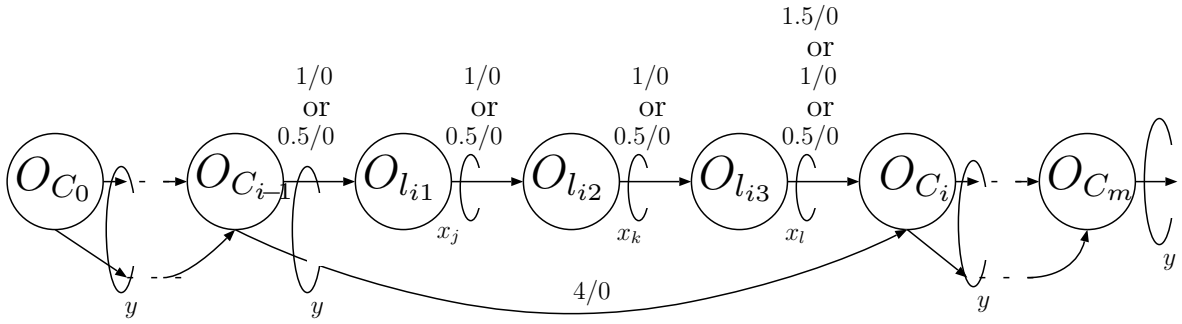


Figure 4.1: A DFG for a 3SAT instance.

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}?"

See Fig.4.1 shows the input instance for (σ, τ)-decision problem transformed from the input instance for 3SAT problem. The description is in following. Data Flow Graph $G = (\mathcal{O}, \mathcal{D})$:

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

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

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

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

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 of negated variables in a clause.

1. For a clause $c_i = (x_j \vee x_k \vee x_l)$; $D^{O_{l_{i1}}}_{y-x_j} = D^{O_{l_{i2}}}_{x_j-x_k} = D^{O_{l_{i3}}}_{x_k-x_l} = 0.5$, $D^{O_{c_i}}_{x_l-y} = 1.5$, $D^{O_{c_i}}_{y-y} = 4$.

2. For a clause $c_i = (x_j \vee x_k \vee \neg x_l)$; $D^{O_{l_{i1}}}_{y-x_j} = D^{O_{l_{i2}}}_{x_j-x_k} = 0.5$, $D^{O_{l_{i3}}}_{x_k-x_l} = D^{O_{c_i}}_{x_l-y} = 1$, $D^{O_{c_i}}_{y-y} = 4$.

3. For a clause $c_i = (x_j \vee \neg x_k \vee \neg x_l)$; $D^{O_{l_{i1}}}_{y-x_j} = D^{O_{l_{i3}}}_{x_k-x_l} = 0.5$, $D^{O_{l_{i2}}}_{x_j-x_k} = D^{O_{c_i}}_{x_l-y} = 1$, $D^{O_{c_i}}_{y-y} = 4$.

4. For a clause $c_i = (\neg x_j \vee \neg x_k \vee \neg x_l)$; $D^{O_{l_{i1}}}_{y-x_j} = D^{O_{c_i}}_{x_l-y} = 1$, $D^{O_{l_{i3}}}_{x_k-x_l} = D^{O_{l_{i2}}}_{x_j-x_k} = 0.5$, $D^{O_{c_i}}_{y-y} = 4$.

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., $0 \leq d^o_{x-y} \leq D^o_{x-y}$.

clk:

We set $clk = 1$.

CS_{spec}:

We set $CS_{spec} = 4m$, where m is the total number of clauses.

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 5 *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 $d^o_{\xi(o')-\xi(o)} \geq 0$ does not affect $\sigma(c^o_{\xi(o)})$.*

Proof of Lemma: 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

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

It means that the hold constraint

$$\begin{aligned} &\sigma(c^o_{\xi(o)}) \cdot clk + \tau(\xi(o)) \\ &\quad \leq \quad \sigma(o^{next(o',\xi(o'))}_{\xi(o')}) \cdot clk + \tau(\xi(o')) + d^o_{\xi(o')-\xi(o)} \end{aligned}$$

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 6 *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 of Lemma: 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.

$$\tau^*(x) = \begin{cases} 0 & \text{if } 0 \leq \tau(x) < 0.5clk \\ 0.5clk & \text{if } 0.5clk \leq \tau(x) < clk \end{cases} \tag{4.1}$$

Note that $\tau(x) \geq \tau^*(x)$ 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

$$\sigma(c_x^o) \cdot clk + \tau(x) \geq \sigma(c_{x'}^{o'}) \cdot clk + \tau(x') + D_{x-x'}^o,$$

we have

$$\begin{aligned} & \sigma(c_x^o) \cdot clk + \tau^*(x) + (\tau(x) - \tau^*(x)) - (\tau(x') - \tau^*(x')) \\ \geq \; & \sigma(c_{x'}^{o'}) \cdot clk + \tau^*(x') + D_{x-x'}^o \end{aligned}$$

Since $\sigma(c_x^o) \cdot clk$, $\tau^*(x)$, $\sigma(c_{x'}^{o'}) \cdot clk$, $\tau^*(x')$ and $D_{x-x'}^o$ 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

$$\sigma(c_x^o) \cdot clk + \tau^*(x) \geq \sigma(c_{x'}^{o'}) \cdot clk + \tau^*(x') + D_{x-x'}^o$$

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 6 and 5, 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.

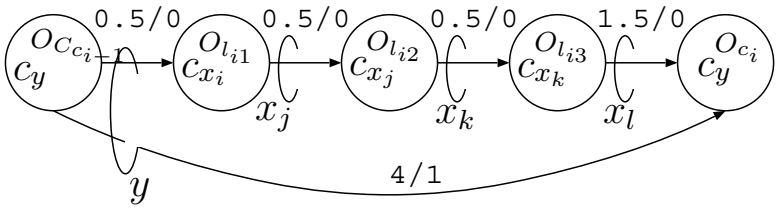
Theorem 4 *(σ, τ)-optimization problem is NP-hard.*

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

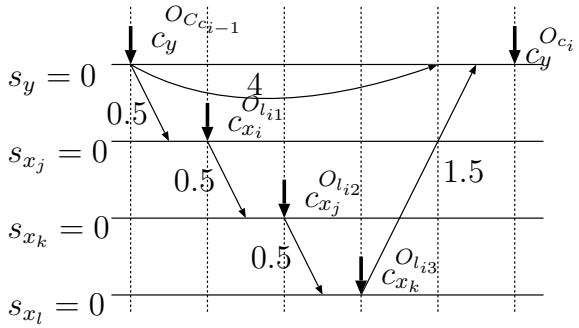
The transformation from an instance of 3SAT ((X, C)) to an instance of the (σ, τ)-decision problem $((G, \rho, \xi, next, D, d, clk, CS_{spec}))$ can be done in polynomial time. We claim that $(G, \rho, \xi, next, D, d, clk)$ has σ and τ such that $\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}})$ is less than or equal to $4m$ if and only if (X, C) is satisfiable. It can be easily verified that $\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}})$ equals to $4m$ if and only if

$$\sigma(c_y^{O_{c_i}}) - \sigma(c_y^{O_{c_{i-1}}}) = 4$$

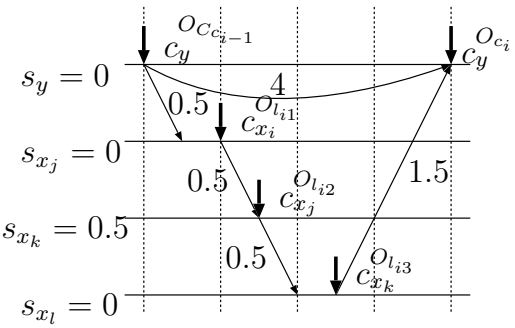
is satisfied for all i, $1 \leq i \leq m$. For a clause $c_i = (x_j \vee x_k \vee x_l)$ (all non-negated variables), $\sigma(c_y^{O_{c_i}}) - \sigma(c_y^{O_{c_{i-1}}}) = 4$ if and only if at least one from $\tau(x_j)$, $\tau(x_k)$ and $\tau(x_l)$ equals to 0.5 (See Fig.4.2), which corresponds to the fact that at least one from x_j, x_k and x_l is assigned 1 and the clause c_i is satisfied. On the other hand, $\sigma(c_y^{O_{c_i}}) - \sigma(c_y^{O_{c_{i-1}}}) = 5$ if we assign 0 to all of $\tau(x_j)$, $\tau(x_k)$ and $\tau(x_l)$, which corresponds to the fact that all of x_j, x_k and x_l are assigned 0 and the clause c_i is not satisfied. The similar arguments are satisfied for the other types of clauses (Fig.4.3-4.5 shows 3 other cases where the clause include negated variables). So, we can conclude that $\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}})$ equals to $4m$ if and only if all clauses are satisfied. □



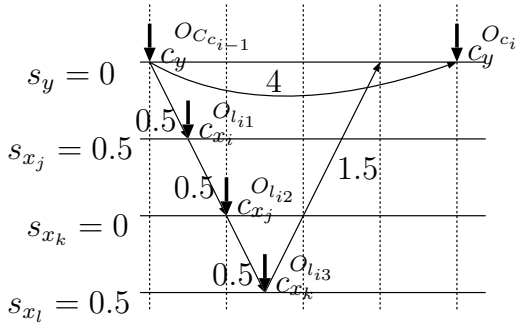
(a) Partial DFG corresponding to clauses $c_i = (x_j \vee x_k \vee x_l)$



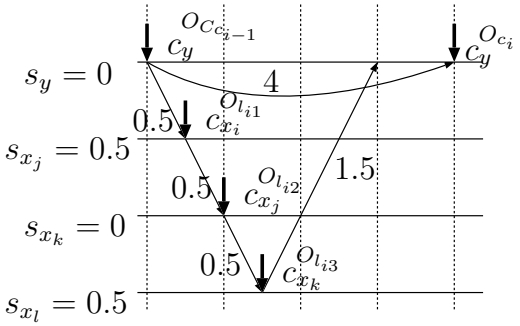
(b) A schedule takes 5 control steps. It corresponds to boolean assignment $(x_j, x_k, x_l) = (0, 0, 0)$ and clauses c_i in not satisfied.



(c) A schedule takes 4 control steps

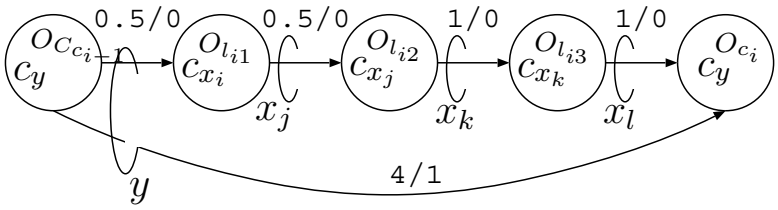


(d) A schedule takes 4 control steps

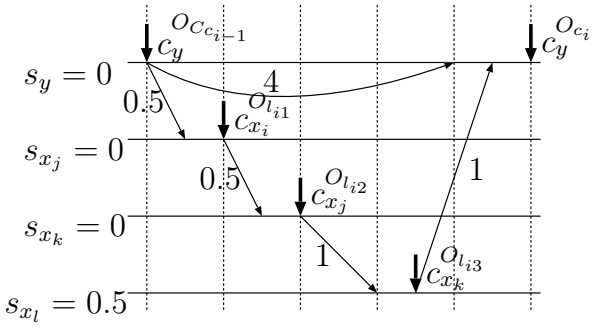


(e) A schedule takes 4 control steps

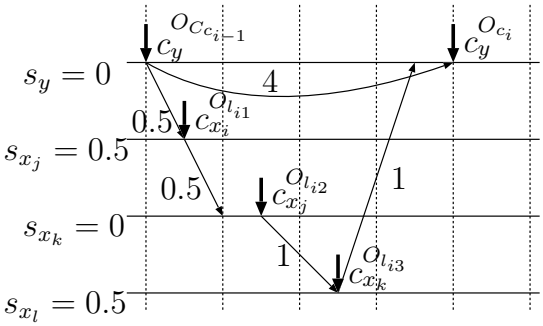
Figure 4.2: 4 control-step assignments for the clause $c_i = (x_j \vee x_k \vee x_l)$.



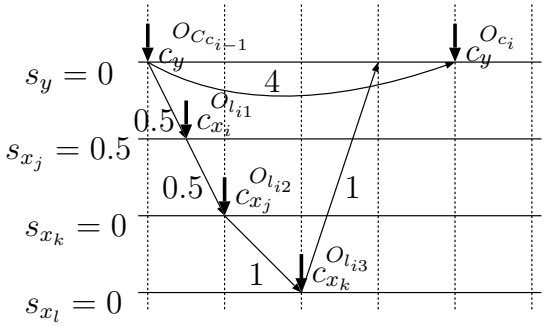
(a) Partial DFG corresponding to clauses $c_i = (x_j \vee x_k \vee \neg x_l)$



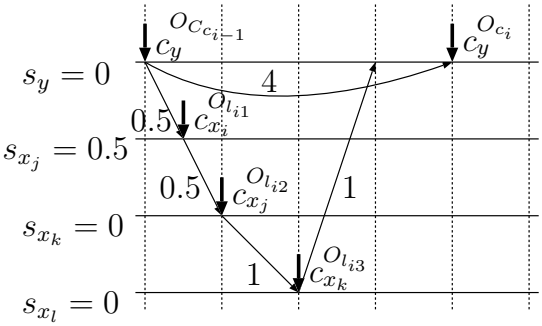
(b) A schedule takes 5 control steps. It corresponds to boolean assignment $(x_j, x_k, x_l) = (0, 0, 1)$ and clauses c_i in not satisfied.



(c) A schedule takes 4 control steps

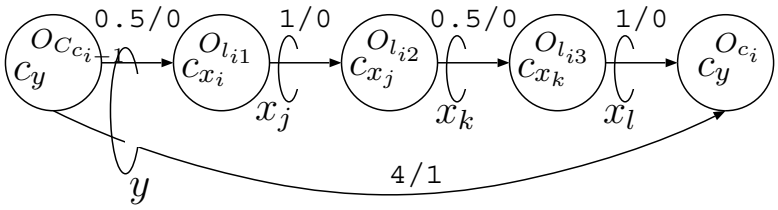


(d) A schedule takes 4 control steps

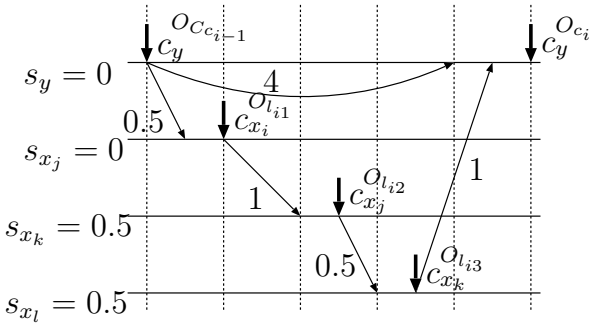


(e) A schedule takes 4 control steps

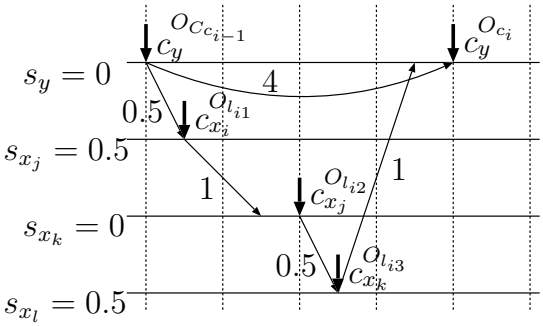
Figure 4.3: 4 control-step assignments for the clause $c_i = (x_j \vee x_k \vee \neg x_l)$.



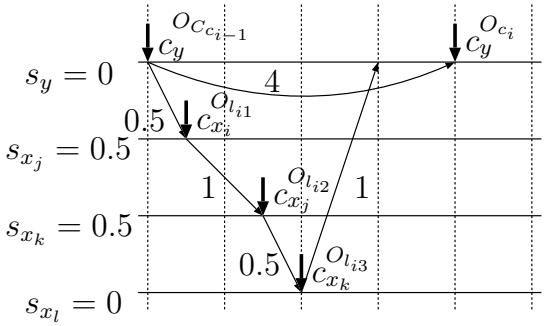
(a) Partial DFG corresponding to clauses $c_i = (x_j \vee \neg x_k \vee \neg x_l)$



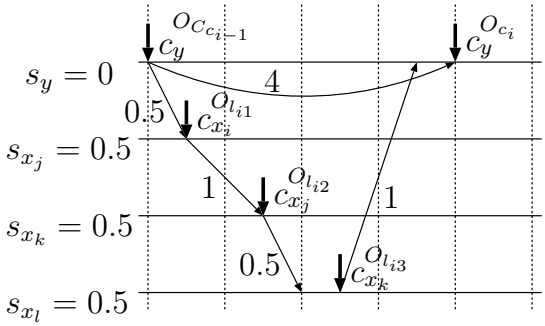
(b) A schedule takes 5 control steps. It corresponds to boolean assignment $(x_j, x_k, x_l) = (0, 1, 1)$ and clauses c_i in not satisfied.



(c) A schedule takes 4 control steps

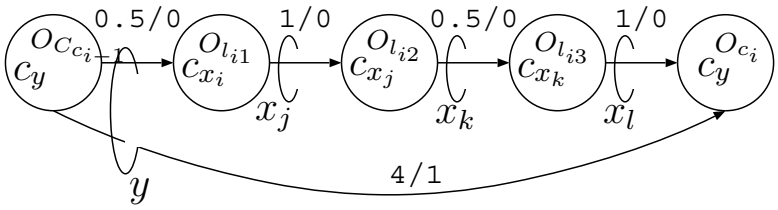


(d) A schedule takes 4 control steps

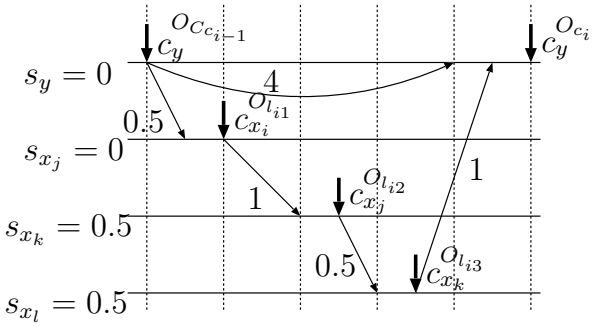


(e) A schedule takes 4 control steps

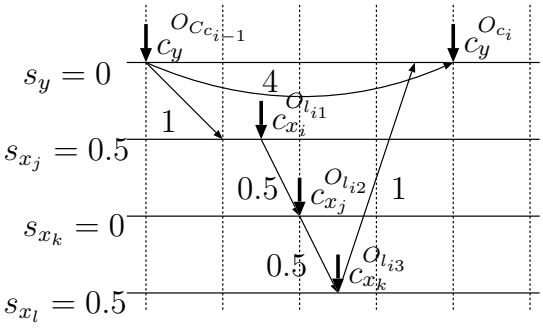
Figure 4.4: 4 control-step assignments for the clause $c_i = (x_j \vee \neg x_k \vee \neg x_l)$.



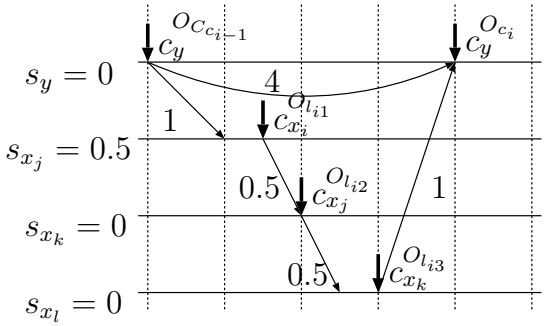
(a) Partial DFG corresponding to clauses $c_i = (\neg x_j \vee \neg x_k \vee \neg x_l)$



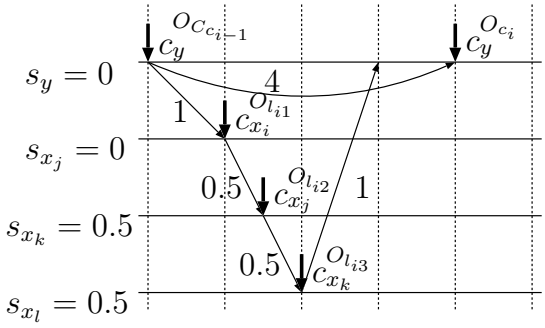
(b) A schedule takes 5 control steps. It corresponds to boolean assignment $(x_j, x_k, x_l) = (1, 1, 1)$ and clauses c_i in not satisfied.



(c) A schedule takes 5 control steps



(d) A schedule takes 5 control steps



(e) A schedule takes 5 control steps

Figure 4.5: 4 control-step assignments for the clause $c_i = (\neg x_j \vee \neg x_k \vee \neg x_l)$.

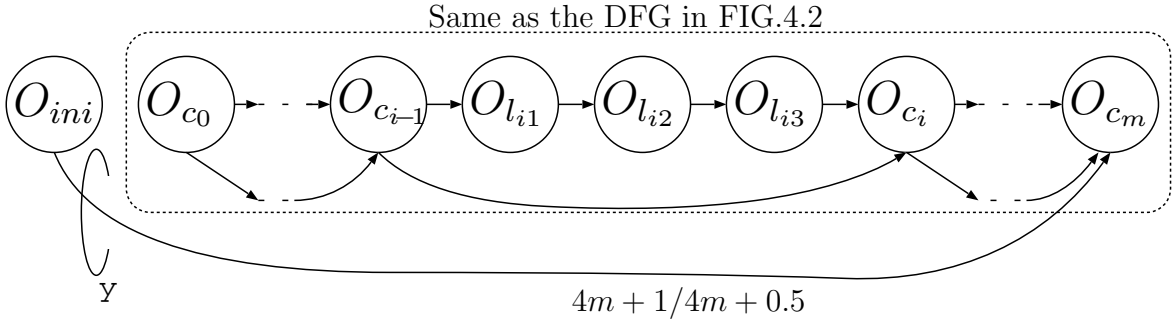


Figure 4.6: A DFG for a 3SAT instance.

4.3 NP-Completeness of (σ,τ)-Solvability Problem

We show that decision problem whether input instance of (σ,τ) problem have a feasible solution or not is a NP-Complete problem. We define the transformation from an instance of 3SAT problem to the solvability decision problem.

Theorem 5 *If and only if there is no positive cycle in G_{rs}, there is a feasible assignment of σ and τ.*

Proof of Theorem:

We modify the instance of (σ,τ)-optimization problem in the proof of Theorem 4.
Data Flow Graph $G' = (\mathcal{O}', \mathcal{D}')$: See Fig.4.6. Add operation O_{ini} and edge (O_{ini}, O_{c_m}) to the DFG in Theorem 4.
Resource assignments ρ' and ξ':

$\xi(O_{ini}) = y$. Other assignments are same as Theorem 4.
Order of operations:

Let $next(y, O_{ini}) = O_{c_0}$.
Maximum path delay D and minimum path delay d:

$D_{y-y}^{O_{c_m}} = 4m + 1$, $d_{y-y}^{O_{c_m}} = 4m + 0.5$.
Others:

Same as Theorem 4. In such a instance, from the edge (O_{ini}, O_{c_m}) in DFG and $next(y, O_{ini}) = O_{c_0}$, following hold constraint holds.

$$\sigma(c_y^{O_{c_m}}) \cdot clk + \tau(y) \leq \sigma(c_y^{O_{c_0}}) \cdot clk + \tau(y) + d_{y-y}^{O_{ini}}$$

So following inequality holds.

$$\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}}) \leq 4m$$

Recall that in optimum σ and τ assignment, if there exists a feasible assignment of X,

$$\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}}) = 4m,$$

otherwise,

$$\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}}) > 4m.$$

So there exists feasible σ and τ assignment which satisfy

$$\sigma(c_y^{O_{c_m}}) - \sigma(c_y^{O_{c_0}}) = 4m$$

if and only if there exists a feasible assignment of X and there exists no feasible σ and τ assignment if if there exists no feasible assignment of X. □

4.4 Conclusion

Simultaneous control step and timing skew assignments is a problem to find σ and τ which satisfy a set of simultaneous inequalities, where each inequality includes two σs and two τs as variables. We have presented that the optimization of σ τ is NP-hard, and solvability of this problem is also NP-complete. The result of this chapter would become an important base for various types of the intentional-skew-aware system optimization problem. Development of an efficient heuristic algorithm for the simultaneous optimization of control schedule and skew assignment is required.

Chapter 5

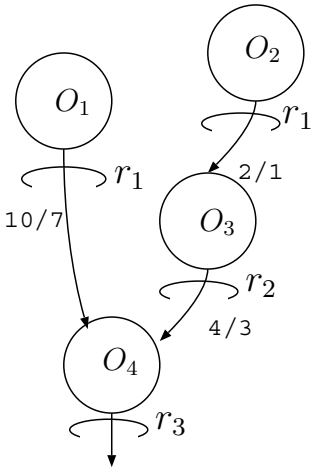
Theorems on Solvability of Simultaneous Optimization of Schedule and Skew assignment

5.1 Introduction

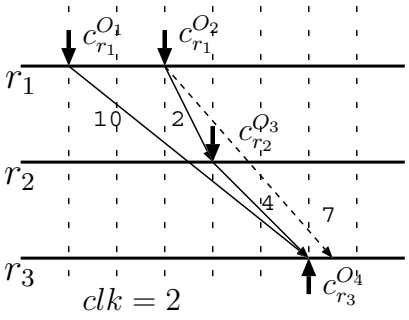
The simultaneous optimization of the control step assignment (σ) and the control skew assignment (τ) is a powerful technique in improving performance. However, the simultaneous optimization problem is a NP-hard problem. This chapter treats an essential and important problem, the solvability of this simultaneous (σ,τ)-problem: to decide whether a given input instance of (σ,τ)-problem has a feasible pair (σ,τ) or not. The feasibility means that σ and τ satisfy all setup and hold inequalities for all operations. If either one of σ and τ is fixed, and the rest is included as variables in each setup or hold inequality, the solvability can be reduced to the problem to test whether a directed graph (skew constraint graph or schedule constraint graph) has a positive cycle or not. It is because, after fixing either one of σ or τ, individual setup/hold inequality includes only two variables, and the problem to determine these variable can be considered as the problem to find the longest path length to each vertex in a directed graph. However, (σ,τ)-problem, each setup/hold inequality includes two σs and two τs as variables, and the solvability of those simultaneous inequalities have been proven to be a NP-Complete problem in Chapter 4.

Theorem 5 suggests that the solvability is intractable in general, and we have to prepare for exponential time algorithm. One other way we can take is tractable sufficient conditions for the solvability. The schedulability under zero skew is one of such tractable sufficient conditions, which can be tested by looking into a schedule constraint graph and searching for a positive cycle in it (if there is a positive cycle, it is unsolvable).

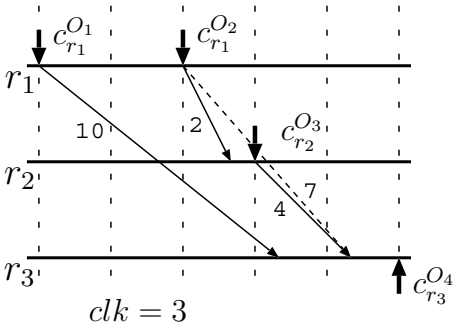
In this chapter, we propose another useful sufficient condition (which can be tested in polynomial time) for the solvability of (σ,τ) problem and it contributes greatly to designing a heuristic algorithm for the simultaneous optimization of (σ,τ). The condition is also a necessary and sufficient condition for an input instance to have feasible solutions for every clock period. In the following, first, skew constraint graph and schedule constraint graph are reviewed, and then modified skew constraint graph will be introduced. After that, our main theorem will be presented.



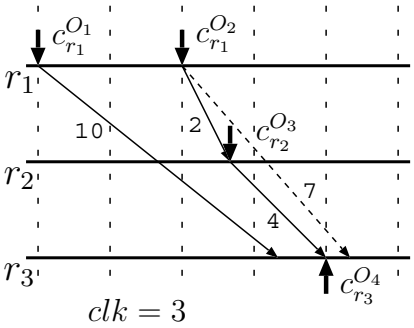
(a) DFG



(b) Timing diagram 1: Schedulable with zero skew for $clk = 2$.



(c) Timing diagram 2: Not schedulable with zero skew for $clk = 3$.



(d) Timing diagram 3: Schedulable schedulable for $clk = 3$ if skew optimization is applied.

Figure 5.1: Schedulability enhancement with skew optimization.

5.2 Motivational Example

Fig.5.1 shows the first example. We assume that the resource binding has been finished, and for each operation, signal path delay from an input register to an output register via a functional unit (and multiplexers if necessary) is given. In general, data path from a multiple-bit register to a multiple-bit register must be 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 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.5.1(a), numbers beside each edge represent the maximum and the minimum delays. The assignment of data to registers is indicated as r_i beside each arc. For example, 10/7 beside an edge (O_1, O_2) represents that the maximum (minimum) path delay from $\xi(O_1) = r_1$ to $\xi(O_4) = r_3$ via a functional unit for the operation O_4 is 10 (7). The order of operations on the register is following; O_1,O_2 for r_1.

If clock period is specified to be 2, we have a feasible zero-skew schedule as shown in Fig.5.1(b). If clock period is 3, there is no feasible schedule under zero skew, but there is only non-zero skew schedule as shown in Fig.5.1(d).

As the second example, we will schedule a DFG shown in Fig.5.3(a). Let orders of operations on the registers are following; O_1,O_2,O_7 for r_1, O_3,O_6 for r_2 and O_4,O_5 for r_3. We can schedule it with clock period 3 as shown in Fig.5.3(b). On the other hand, with clock period 4, we have three inequalities.

$$\tau(r_2) \leq \tau(r_3) + 4k_1 \leq \tau(r_2) + 1$$

$$\tau(r_1) + 2 \leq \tau(r_3) + 4k_2 \leq \tau(r_1) + 3$$

$$\tau(r_1) + 2 \leq \tau(r_2) + 4k_3 \leq \tau(r_1) + 3$$

from nodes $\{O_1,O_2,O_3,O_4\}$ in DFG and we also have

$$\tau(r_3) + 2 \leq \tau(r_2) + 4k_4 \leq \tau(r_3) + 3$$

$$\tau(r_3) + 2 \leq \tau(r_1) + 4k_5 \leq \tau(r_3) + 3$$

$$\tau(r_2) \leq \tau(r_1) + 4k_6 \leq \tau(r_3) + 1$$

from nodes $\{O_4,O_5,O_6,O_7\}$ where $k_1, k_2, k_3, k_4, k_5, k_6$ are auxiliary integers. We cannot fulfill all inequalities simultaneously, so we cannot schedule with clock period 4. Let orders of operations on the registers are following; O_2,O_1,O_7 for r_1, O_3,O_6 for r_2 and O_4,O_5 for r_3. We can schedule it with clock period 4 as shown in Fig.5.3. The schedulable in Fig.5.3 is more excellent than the schedulable in Fig.5.3(b) in both computational time and size of the controller (the number of control steps). If we decide the resource assignment and the operation order so that it is schedulable with zero skew and then optimize schedule and skew assignment, we might miss such an excellent solution. So another condition is required.

It is clear that the schedulability depends on a resource assignment, execution order of operations, path delays and a clock period. Our question tackled in this paper is; "Is it possible to decide whether a given instance has a feasible solution or not? How to decide it?"

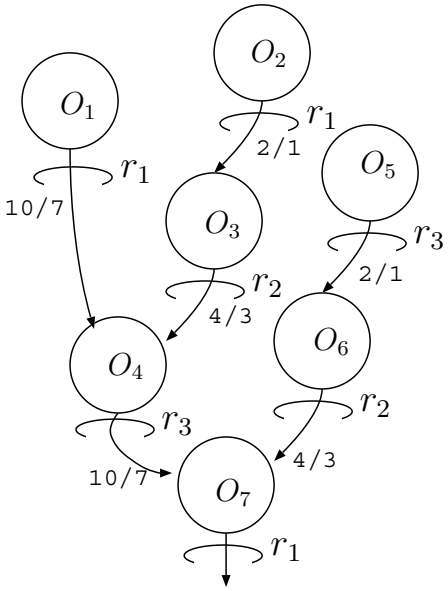
5.3 Theorems on Solvability of (σ,τ)-Optimization Problem

5.3.1 Skew Constraint Graph

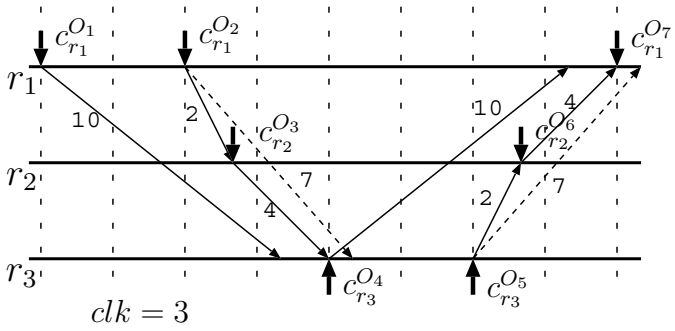
From (2.2.1)-(2.2.1), we have

$$\tau_r - \tau_m \geq (\sigma(c_m^{o_p}) - \sigma(c_r^{o_k})) \cdot clk + \tau_{err} + D_{m-r}^{o_k} + s, \quad (5.1)$$

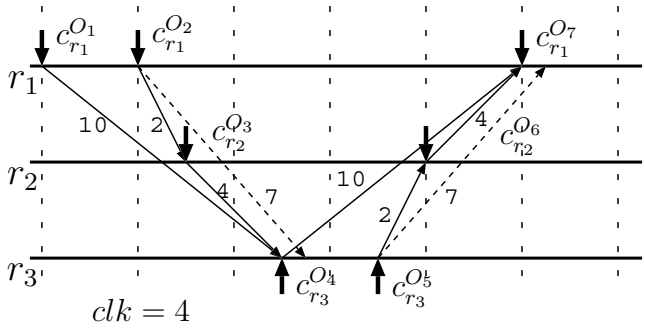
$$\tau_m - \tau_r \geq (\sigma(c_r^{o_k}) - \sigma(c_m^{next(m,o_p)})) \cdot clk + \tau_{err} - d_{m-r}^{o_k} + h. \quad (5.2)$$



(a) DFG



(b) Timing diagram 1: Schedulable with skew optimization for $clk = 3$



(c) Timing diagram 2: Not schedulable for $clk = 4$

Figure 5.2: A non-schedulable example of DFG, resource assignment and operation order with skew optimization.

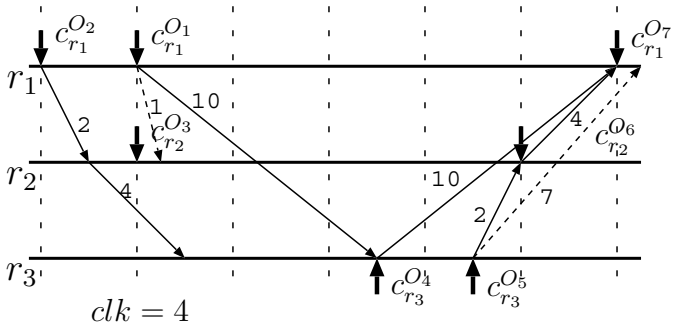


Figure 5.3: A schedulable order for DFG and resource assignment in Fig..

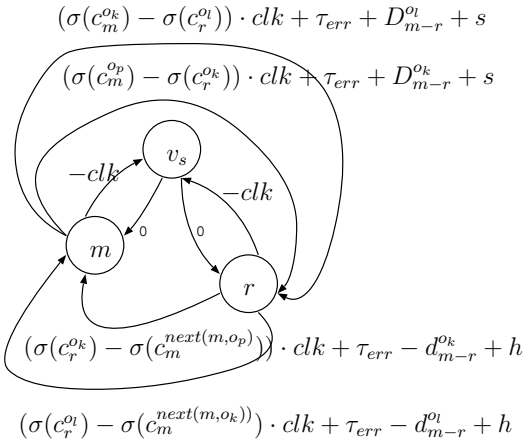


Figure 5.4: Skew constraint multigraph

We generate a skew constraint multigraph $G_\tau = (V_\tau, E_\tau)$ from (5.1-5.2) as shown in Fig.5.4. V_τ is a set of multiplexers, registers and one auxiliary source node v_s. A set of weighted edges E_τ is the union of a set of edges reflecting (5.1) or (5.2) (i.e., an edge (m, r) with weight $(\tau_m^{o_p} - \tau_r^{o_k}) \cdot clk + \tau_{err} + D_{m-r}^{o_k} + s$ or an edge (r, m) with weight $(\tau_r^{o_k} - \tau_m^{next(m,o_p)}) \cdot clk + \tau_{err} - d_{m-r}^{o_k} + h)$ over all operations, and a set of auxiliary edges $\{(m, v_s)|m \in V_\tau \setminus v_s\} \cup \{(v_s, m)|m \in V_\tau \setminus v_s\}$. Edge weights for $\{(m, v_s)|m \in V_\tau \setminus v_s\}$ and $\{(v_s, m)|m \in V_\tau \setminus v_s\}$ are $-clk$ and 0, respectively. Then, skew assignment problem is now considered as the problem to assign real values to vertices in G_τ, and the maximum path length from v_s to each vertex gives us a solution, i.e., skew of each register and multiplexer. If G_τ has a positive cycle, feasible skew schedule does not exist.

Once we assign a value to clk and τ, multi-edges between a pair of vertices can be reduced to one edge which has maximum weight (since only maximum path length is our concern), and hence we can use Ford algorithm to compute maximum path length.

5.3.2 Schedule Constraint Graph

From (2.2.1)-(2.2.1) with regarding integral σ, we have

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

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

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 (5.3.2) or (5.3.2), 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 CS: the number of steps for the application. A path which gives CS is called a critical path. If G_σ has a positive cycle, feasible schedule does not exist.

5.3.3 Modified Skew Constraint Graph

Modified skew constraint graph $G_m(V_m = V_\tau, E_m)$ is a subgraph of a skew constraint graph. The edge set E_m is a subset of E_τ. If and only if the edge of schedule constraint graph $(c_x^o, c_{x'}^{o'})$ is an element of a directed cycle, corresponding (expressing same constraint inequality) edge (x, x') in skew constraint graph is an element of E_m. Edge weight of (x, x') is $\tau_{err} + D_{x-x'}^{o'} + s$ if the weight in skew constraint graph is $(\sigma(c_x^o) - \sigma(c_{x'}^{o'})) \cdot clk + \tau_{err} + D_{x-x'}^{o'} + s$ or $\tau_{err} - d_{x-x'}^{o'} + h$ if weight in skew constraint graph is $(\sigma(c_{x'}^{o'}) - \sigma(c_x^{next(o)})) \cdot clk + \tau_{err} - d_{x-x'}^{o'} + h$.

Theorem 6 *If and only if there is no positive cycle in modified skew constraint graph, the input instance has feasible control-step and skew assignments for every clock period.*

Before going to the proof of the theorem, we show some preliminary lemmas.

Lemma 7 *Let D_{max} be the maximum value of path delays. If there is a cycle C in schedule constraint graph and $c_x^o, c_{x'}^{o'} \in C$, then*

$$\left|(\sigma(c_x^o)clk + \tau(x)) - \left(\sigma(c_{x'}^{o'})clk + \tau(x')\right)\right| < |\mathcal{M}|D_{max}$$

Proof of Lemma: Let $P1$ is a path from c_x^o to $c_{x'}^{o'}$ in C. $D_\sigma(e)$ denote the weight of edge e except two terms which contain τ. Summing up constraint inequalities along $P1$, we obtain

$$\left(\sigma(c_{m'}^{o'})clk + \tau(m')\right) - \left(\sigma(c_m^0)clk + \tau(m)\right) > \sum_{e \in P_1} D_\sigma(e).$$

So we have

$$\left(\sigma(c_m^0)clk + \tau(m)\right) - \left(\sigma(c_{m'}^{o'})clk + \tau(m')\right) < -\sum_{e \in P_1} D_\sigma(e) < |\mathcal{M}|D_{max}.$$

Similar discussion about path from $c_{x'}^{o'}$ to c_x^o in C leads the substance of the lemma. □

Lemma 8 *If there is a feasible control-step and skew assignment for sufficiently large clock period, there is a control-step assignment which assign all control signals belonging to the same cycle to the same control-step.*

Proof of Lemma: Assume that there exists control-step assignment σ_{init}. Compute skew assignment with skew constraint graph. Skew value $\tau(m)$ for $m \in \mathcal{M}$ can be expressed as $n\ clk + \sum_{e \in P(m)} d(e)$ where $P(m)$ is a longest path to m from the source vertex v_s in skew constraint graph and $d(e)$ is the weight of edge e other than σ (i.e. the weight of e in modified skew constraint graph). Following inequality is obvious.

$$-|\mathcal{M}|D_{max} \leq \sum_{e \in P(m)} d(e) \leq |\mathcal{M}|D_{max}$$

Recall $0 \leq \tau(m) < clk$ and let clk be $clk > 3|\mathcal{M}|D_{max}$. Then one of followings holds.

$$\tau(m) \leq |\mathcal{M}|D_{max}$$

$$clk - |\mathcal{M}|D_{max} \leq \tau(m) < clk$$

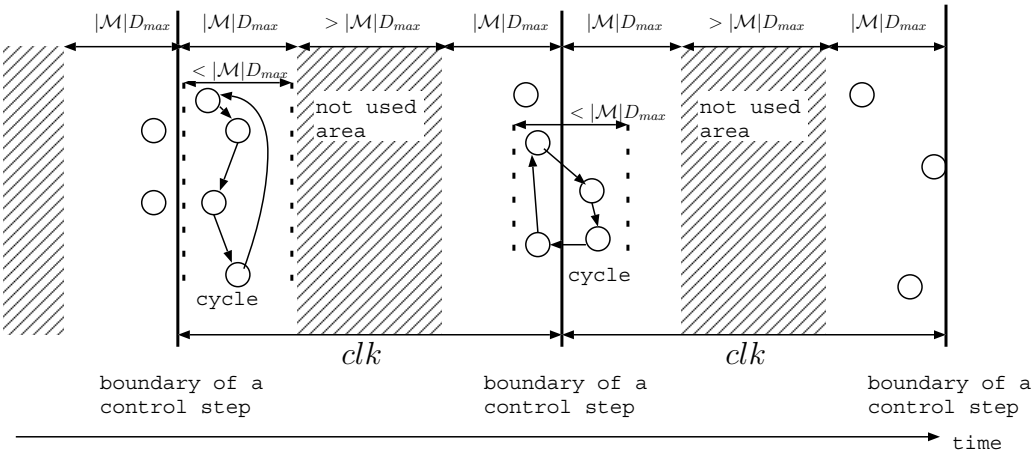
When we put off the timing of control signals by $|\mathcal{M}|D_{max}$ and let new control-step and skew assignment be σ' and τ', we have

$$\tau'(m) \leq 2|\mathcal{M}|D_{max}$$

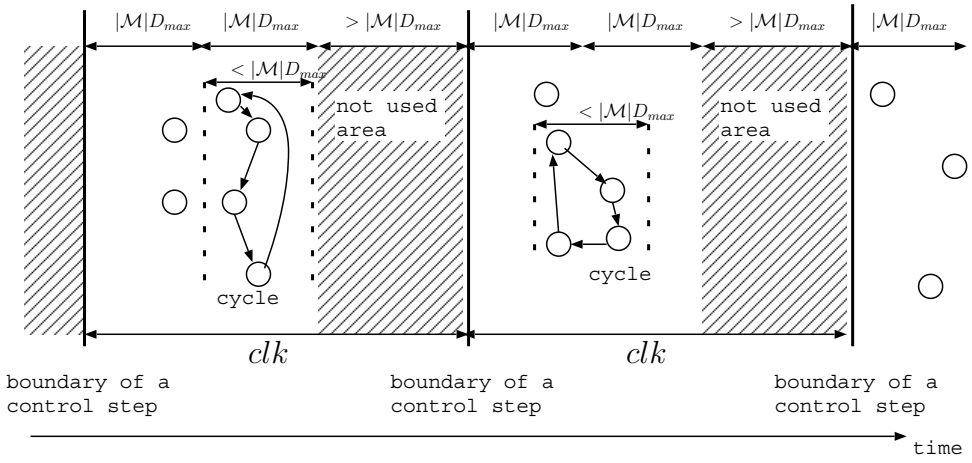
(See Fig. 5.5. Circles in the figure show arrival timing of control signals.). If there exist a cycle C and $\sigma'(c_m^0), \sigma'(c_{m'}^{o'}) \in C$ such that $\sigma'(c_m^o) \neq \sigma'(c_{m'}^{o'})$, the inequality

$$\left|\left(\sigma'(c_m^0)clk + \tau(m)\right) - \left(\sigma'(c_{m'}^{o'})clk + \tau(m')\right)\right| > |\mathcal{M}|D_{max}$$

holds and it is the contradiction to Lemma7. So σ' is a control-step assignment such that all control signals belonging to the same cycle is assigned to the same control-step. □



(a) Original timing.



(b) All cycles in one control step.

Figure 5.5: Timing shift of control signals.

Lemma 9 *If there is no positive cycle in modified skew constraint graph, the longest path length for each vertex is a feasible skew value for clock period clk with* $clk > 3|\mathcal{M}|D_{max}$.

Proof of Lemma: If an edge in skew constraint graph is not an element of any cycle, the skew assignment to its end points does not affect whether there is a positive cycle or not. So if we can modify control-step assignment, we don't have to consider a corresponding edge in skew constraint graph. If there is a feasible control-step and skew assignment for sufficiently large clock period, there is a control-step assignment which assigns all control signals belonging to same cycle to same control-step. For such control-step assignment, two σ values in a constraint inequality is the same value i.e. edge weight of skew constraint graph is the same as modified skew constraint graph. So, modified skew constraint graph gives a feasible skew assignment for $clk > 3|\mathcal{M}|D_{max}$. □

Lemma 10 *If there exists a feasible control-step and skew assignments for all clock period larger than certain clock period* clk_1*, there is feasible solution for every clock period.*

Proof of Lemma: For any clock period $clk < clk_1$, there exists a positive integer k such that $kclk > clk_1$. Let σ and τ be a control-step and skew assignment for clock period $kclk$. Then, the following σ' and τ' are feasible control-step and skew assignments.

$$\sigma'(c_x^o) = k\sigma(c_x^o) + \lfloor \tau(x)/clk \rfloor$$

$$\tau'(x) = \tau(x) - \lfloor \tau(x)/clk \rfloor$$

□

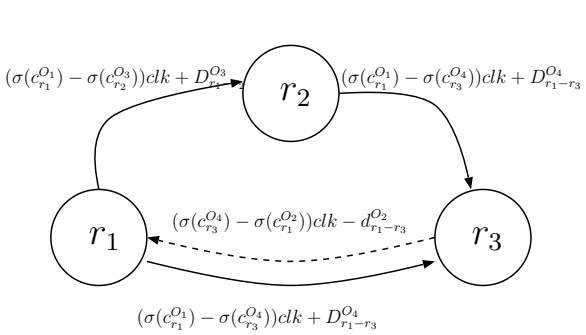
Proof of Theorem 1: Lemma 8 implies necessity of the condition. Sufficiency is obvious from Lemma 9 and 10. □

5.4 Examples

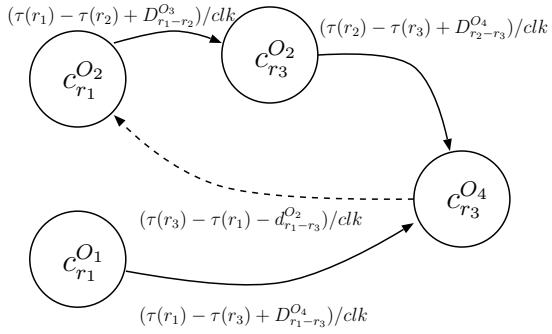
We show some examples of modified skew constraint graph. Fig.5.6(a) shows the skew constraint graph for the DFG, resource assignment and operation order in Fig.5.1. The schedule constraint graph is shown in Fig.5.6(b). There is a cycle $c_{r_1}^{O_2}, c_{r_2}^{O_3}, c_{r_3}^{O^4}$. So the modified schedule graph (Fig.5.6(c)) have edges corresponding to $(\tau(r_1) - \tau(r_2) + D_{r_1-r_2}^{O_3})/clk$, $(\tau(r_2) - \tau(r_3) + D_{r_2-r_3}^{O_4})/clk$ and $(\tau(r_3) - \tau(r_1) - d_{r_1-r_3}^{O_2})/clk$. The modified constraint graph have no positive cycles, so we can schedulable the DFG, resource assignment and operation order in Fig.5.1 for any clock period if we optimize skew assignment.

Next example in Fig.5.8 is corresponding to Fig.5.3. The schedulable constraint graph (Fig.5.8(b)) have two cycles $c_{r_1}^{O_2} c_{r_3}^{O_2} c_{r_3}^{O_4}$ and $c_{r_3}^{O_5} c_{r_2}^{O_6} c_{r_1}^{O_7}$. The modified skew constraint graph consists of these 6 edges and have a positive cycles $r_1 r_2$ and $r_2 r_3$. So we can not schedulable the DFG, resource assignment and operation order in Fig.5.3 for some clock period even if we optimize skew assignment.

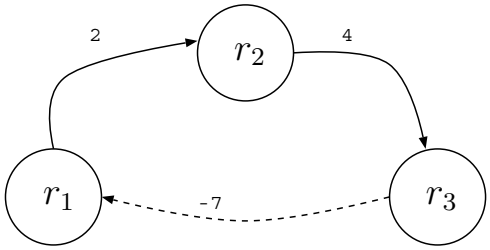
If we change the order of O_1 and O_2 on r_1, the cycle $c_{r_1}^{O_2} c_{r_3}^{O_2} c_{r_3}^{O_4}$ is resolve in schedulable constraint graph (**??**(b)) and the modified constraint graph have no positive cycle (**??**(c)). So we can schedule the DFG, resource assignment and operation order in Fig.5.3 for any clock cycle.



(a) The skew constraint graph



(b) The schedule constraint graph

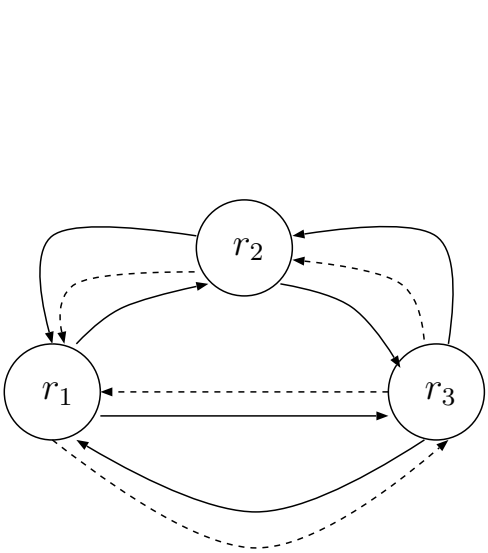


(c) The modified skew constraint graph

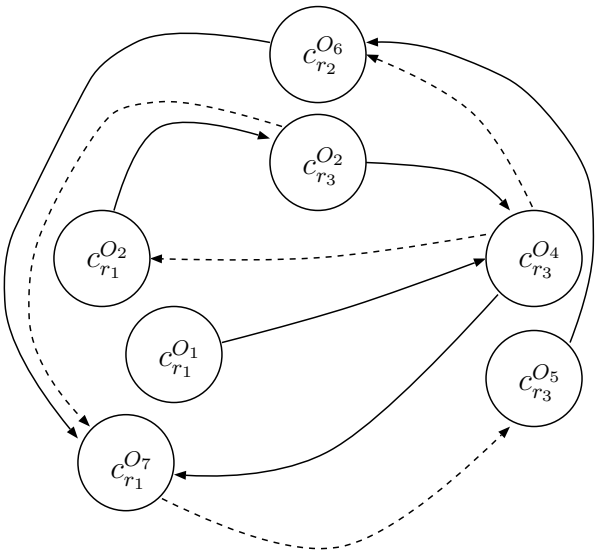
Figure 5.6: Modified skew constraint graph corresponding to Fig.5.1

5.5 Conclusions

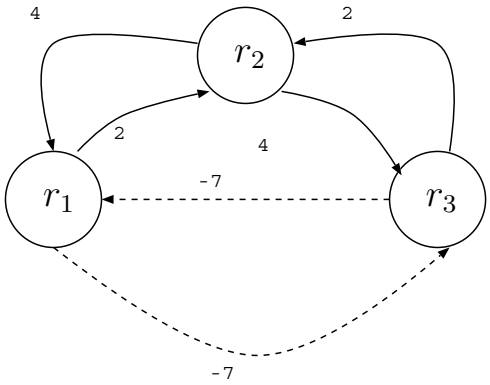
Simultaneous control step and timing skew assignments is a problem to find σ and τ which satisfy a set of simultaneous inequalities, where each inequality includes two σs and two τs as variables. We have shown a non-trivial sufficient condition for an input instance to have a feasible solution. It is useful in a practical design, because the condition is now a necessary and sufficient condition for an input instance to have a feasible solution for every choice of clock period.



(a) The Skew Constraint graph

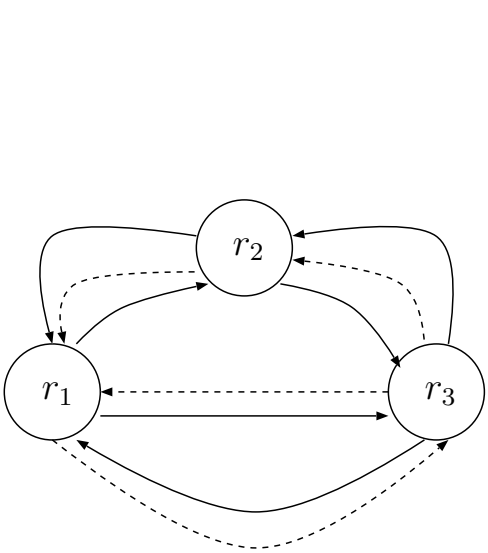


(b) The schedule Constraint graph

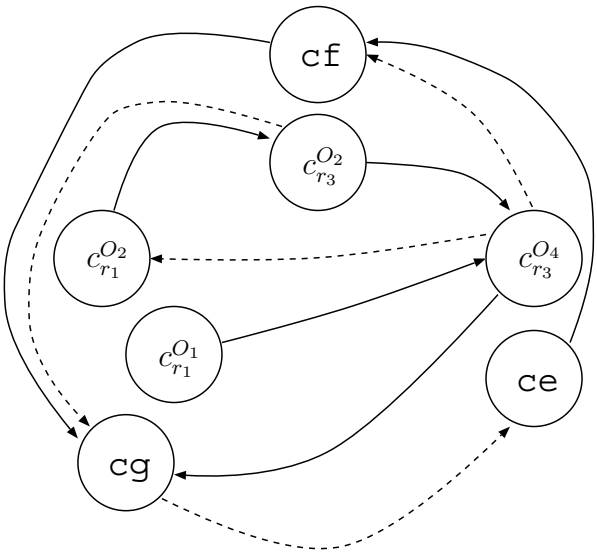


(c) The Modified Skew Constraint graph

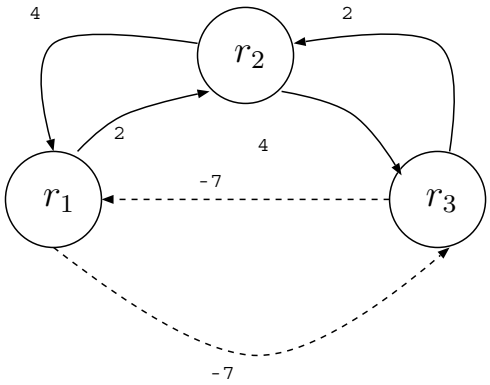
Figure 5.7: Modified skew constraint graph corresponding to Fig.5.3



(a) The Skew Constraint graph



(b) The schedule Constraint graph



(c) The Modified Skew Constraint graph

Figure 5.8: Modified skew constraint graph corresponding to Fig.5.3

Chapter 6

Heuristic Algorithm for Simultaneous Optimization of Schedule and Skew assignment

6.1 Introduction

The skew of control signals can be utilized intentionally for improving system performances. In conventionally, skew is utilized only for reducing clock period. In this chapter, we focus to reduce the number of control steps (makespan) for a target application. (σ,τ)-Optimization problem is a problem to decide control step assignment of control signals and skew assignment to modules so as to minimized the number of control steps under given resource binding, order between operations which bound to the same resource, delays and clock period. In other words, we have to decide σ and τ so as to satisfy setup and hold constraint (2.2.1)-(2.2.1) and minimize $\max_{s \in \mathcal{S}} \sigma(s)$.

The simultaneous optimization of skew and control step assignment of control signals is a NP-hard problem. It implies that there is no efficient solution algorithm. So, in this chapter, we show a heuristic algorithm for this (σ, τ)-optimization problem.

6.2 Motivational Example

Fig.6.1(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 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.6.1(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.

Fig.6.1(b) shows a schedule of 6 control signals $c_{r_1}^{o_1}$, $c_{r_2}^{o_2}$, $c_{r_3}^{o_3}$, $c_{r_1}^{o_4}$, $c_{r_2}^{o_5}$, and $c_{r_3}^{o_6}$. 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.6.1(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.6.1(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.6.1(d). Fig.6.1(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.6.1(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.6.1(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.6.1(e)), the skew assignment and the control step assignment must be treated simultaneously.

6.3 Heuristic Algorithm for (σ, τ)-optimization Problem

Our heuristic algorithm uses skew constraint graph G_τ (see Section 5.3.1) and schedule constraint graph G_σ (see Section 5.3.2).

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 "$(t_{err} + D^{o_j}_{x_i - r_j} + s) \bmod clk$" or "$(t_{err} - d^{o_j}_{x_i - r_j} + h) \bmod clk$" 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 $(t_{err} + D^{o_j}_{x_i - r_j} + s) \bmod clk$ or $(t_{err} - d^{o_j}_{x_i - r_j} + h) \bmod clk$ depending on the edge weight.

On the other hand in G_σ, since the right hand side of (5.3.2)-(5.3.2) has a ceiling, if the relative skew $(\tau(r_j) - \tau(x_i)) \bmod clk$ is equal to $(t_{err} + D^{o_j}_{x_i - r_j} + s) \bmod clk$ or $(t_{err} - d^{o_j}_{x_i - r_j} + h) \bmod clk$, the weight of the edge reflecting inequality (5.3.2) or (5.3.2) is minimized. Therefore, to minimize CS, it is efficient to set relative skew to $(t_{err} + D^{o_j}_{x_i - r_j} + s) \bmod clk$ or $(t_{err} - d^{o_j}_{x_i - r_j} + h) \bmod clk$ 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.6.3. We start with the spanning tree T whose edge set is $\{(v_s, m) | m \in V \setminus v_s\}$ 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 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. 6.2 shows the replacement of edges. We add (u, v) to T only if u and v belong to different partite sets.

6.4 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 binding, 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 binding 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 6.1 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.

Figures 6.4 through 6.6 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 binding, 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-valued schedule, which is introduced in Section 2.2.1. 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.

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 Fig.6.7 through 6.12. Design points given by "skew after schedule" are the result of the two-step

Table 6.1: 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

algorithm; scheduling (with zero skew) followed by skew optimization. Figures 6.7, 6.8, and 6.9 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 6.10, 6.11, and 6.12 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).

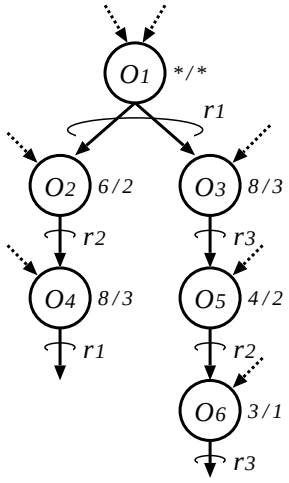
6.5 Conclusion

We have introduced a novel optimization problem, simultaneous schedule (control step assignment) and skew optimization problem. We presented a heuristic algorithm for the simultaneous control step and skew optimization under given clock period. It start with initial skew assignment (zero for all modules) and modify a skew assignment so as to minimize the length of the longest path in schedule constraint graph. So the algorithm cannot compute a solution for the instance which have no feasible schedule with zero

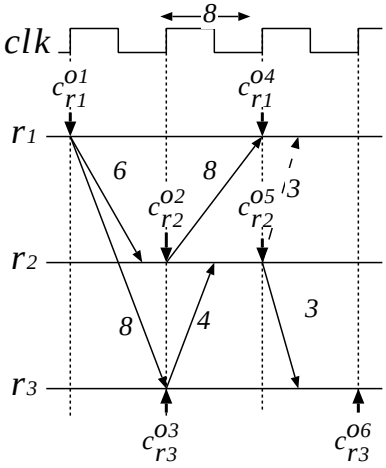
skew.

The experimental result shows the effectiveness of simultaneous optimization of schedule and skew and efficiency of our algorithm.

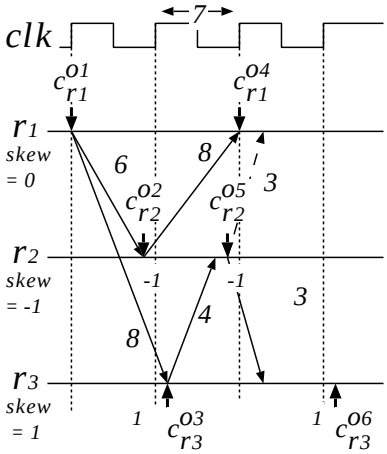
The algorithm has the potential to play a central role in various scenarios of skew-aware RT level synthesis. A study of the relation between the simultaneous optimization of skew and re-timing in logic level and our problem in RT level is one of the interesting future works.



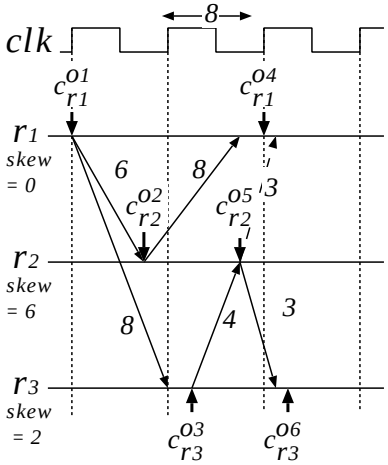
(a) DFG



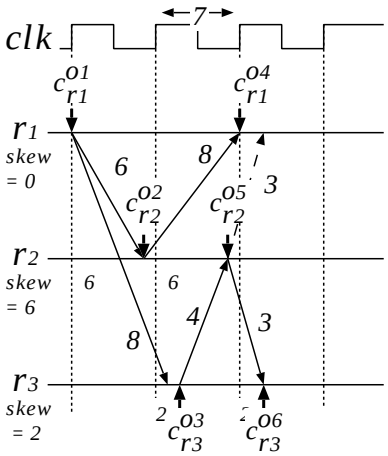
(b) Design 1



(c) Design 2



(d) Design 3



(e) Design 4

Figure 6.1: Several different types of skew aware designs

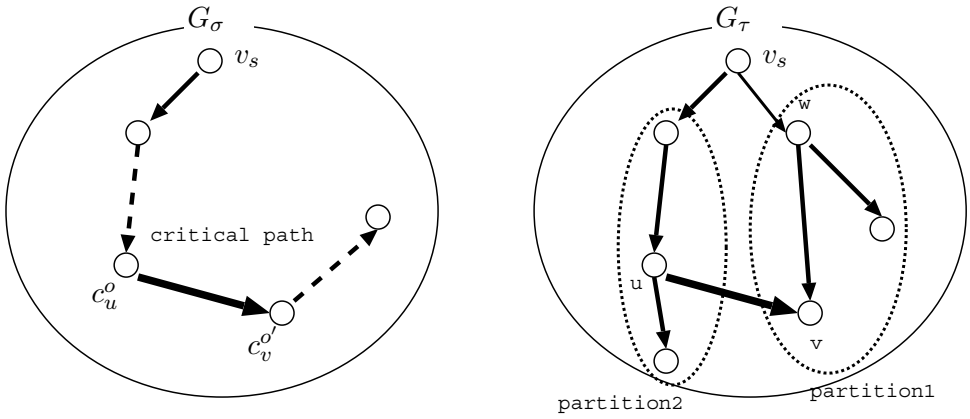


Figure 6.2: We add an edge on a critical path in G_σ to T.

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.

Figure 6.3: Heuristic algorithm

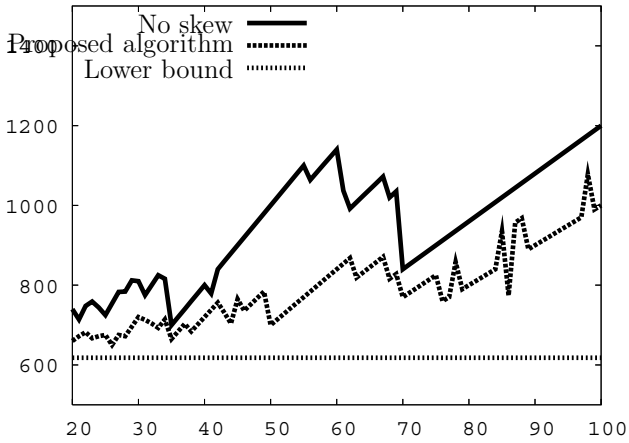


Figure 6.4: Application time ($CS \times clk$) vs. clk for Jaumann

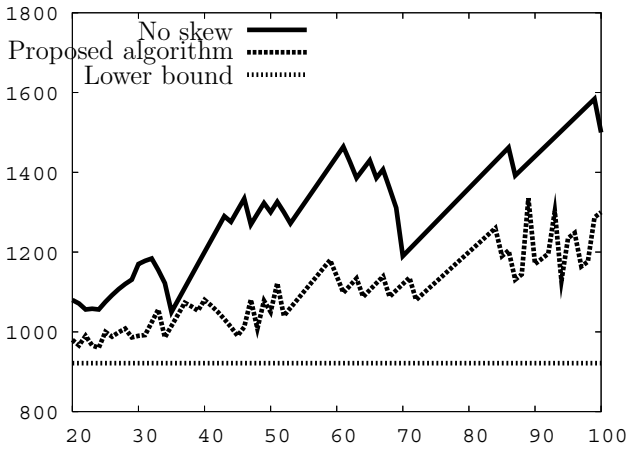


Figure 6.5: Application time ($CS \times clk$) vs. clk for Lattice

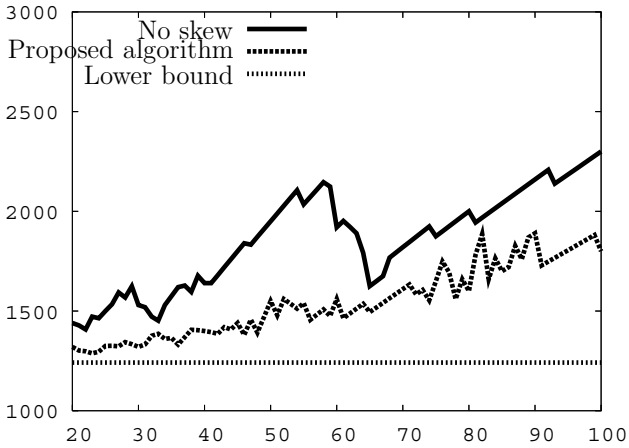


Figure 6.6: Application time ($CS \times clk$) vs. clk for Elliptic

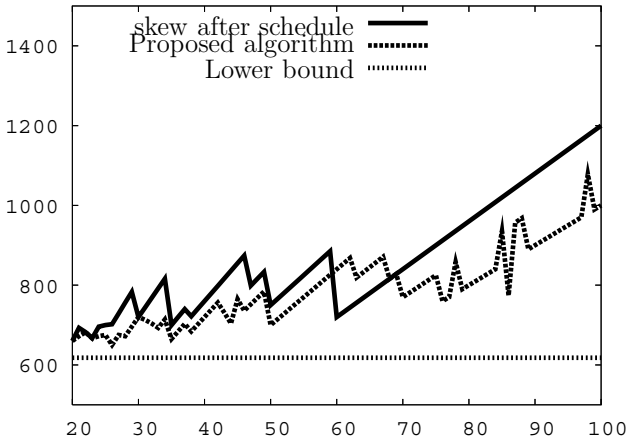


Figure 6.7: Application time ($CS \times clk$) vs. clk for Jaumann

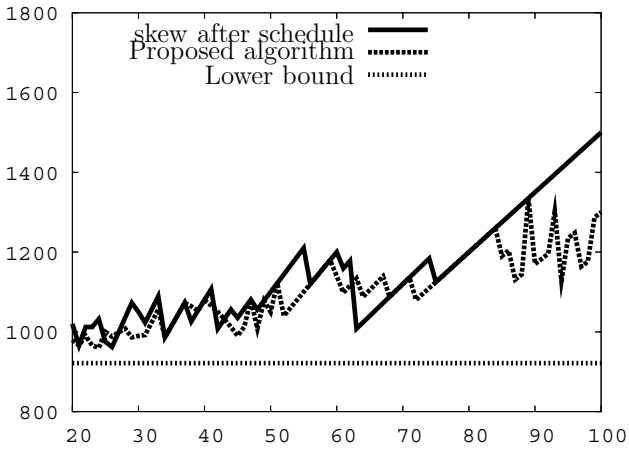


Figure 6.8: Application time ($CS \times clk$) vs. clk for Lattice

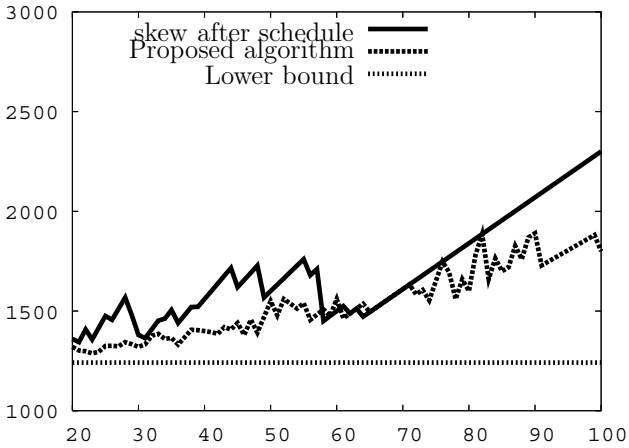


Figure 6.9: Application time ($CS \times clk$) vs. clk for Elliptic

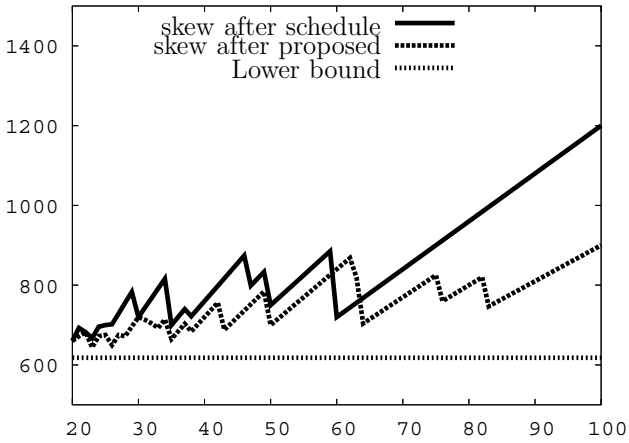


Figure 6.10: Application time ($CS \times clk$) vs. clk for Jaumann

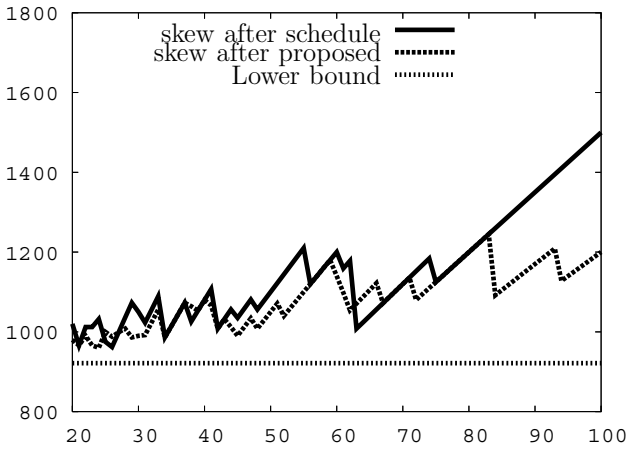


Figure 6.11: Application time ($CS \times clk$) vs. clk for Lattice

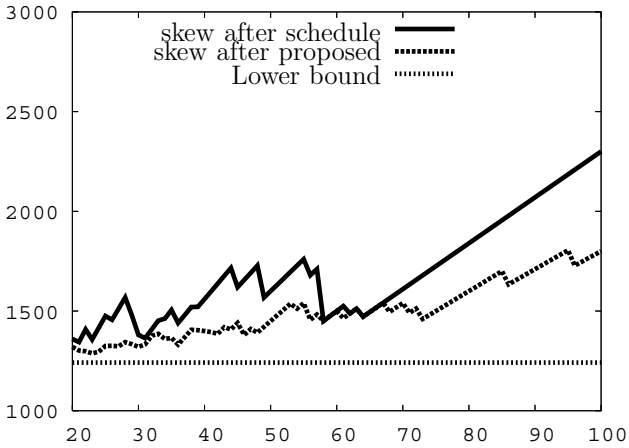


Figure 6.12: Application time ($CS \times clk$) vs. clk for Elliptic

Chapter 7

Conclusion

In this paper, we have introduced a new problem, the skew of the arrival time of control signals to multiplexers and registers in a RT level datapath. We have investigated about simultaneous optimization of skew assignment to control signals to registers and registers, control step assignment of control signals, and clock period. As a first contribution, we have proven the NP-hardness of simultaneous optimization of three subjectives by induction from 3SAT problem in Chapter 3. Based on the results, we focused on sub problems: (1) simultaneous skew and clock period optimization under given control step assignment of control signals and (2) simultaneous skew and control step assignment optimization under given clock period.

For (1), we have presented an approximation algorithm in Section 2.3. The algorithm is based on combination of the longest path algorithm on skew constraint graph and binary search algorithm. The algorithm compute near optimal solution in shorter time than simplex algorithm. The ratio of computation time expands as the scale of input instance grows. The first experiment in Section 2.3.2 shows the effectiveness of skew optimization. It reduced clock period by 74% in average. The second experiment i.e. high-level synthesis with skew optimization shows an importance to optimize control schedule with regarding internal skew.

For (2), we have proven that the optimization is NP-hard in Chapter 4. We have also proven that the decision problem whether there exists a feasible solution is a NP-complete problem. These result implies that there is no efficient solution algorithm unless $\mathcal{P} = \mathcal{NP}$. It also suggests the need for a heuristic algorithm. So we have focused to the conditions for the solvability in Chapter 5. As a result, we have proven that a sufficient and necessary condition for an input instance (a input algorithm, resource assignment, order of operation on a resource and delay information) to have a feasible solution for any clock period is that there is no positive cycle in modified skew constraint graph. The condition is also a sufficient condition for the input instance have a feasible solution for a specific clock period. Then we developed a heuristic algorithm in which we construct an initial solution (assign zero skew to all modules) and improve the skew assignment. To break the limitation of the algorithm that we cannot compute the solution for the instance which have no feasible control schedule with zero skew, it will useful to utilize the sufficient condition presented in Chapter 5 to generate an initial skew.

After all, we have developed a solution algorithm for two problems: skew assignment optimization so as to minimize the clock period under given control step assignment and control step assignment so as to minimize the number of control steps under given clock

period.

Intentional skew control is a promising key technology not only to improve VLSI performance, but also to provide tunability for each VLSI to operate with its own maximum performance, which may overcomes the current and future process variability problem. Those results presented in this paper should be important theoretical base of skew-aware datapath design.

As a future directions, a heuristic algorithm to optimize all of control step assignment, skew assignment, clock period is required. The order of operation on a resource should also be a subjecives of optimization. Especially, the generation of an order which satisfy the sufficient condition in Chapter 5 or explicit condition (there is a feasible schedule with zero skew) to solvability is an important problem. 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.

Bibliography

[1] J. P. Weng, A. C. Parker, "3D scheduling: high-level synthesis with floorplanning," Proc. Design Automation Conf., pp.668-673, 1991.

[2] Y. M. Fang, D. F. Wong, "Simultaneous functional unit binding and floorplanning," Proc. Int. Conf. on Computer Aided Design, pp.317-321, 1994.

[3] V. G. Moshnyaga, K. Tamaru, "A placement driven methodology for high-level synthesis of sub-micron ASIC's," Proc. Int. Symp. on Circuits and Systems, vol. 4, pp572-575, 1996.

[4] S. Tarafdar, M. Leeser, Z. Yin, "Integrating floorplanning in data-transfer based high-level synthesis," Proc. Int. Conf. on Computer Aided Design, pp.412–417, 1998.

[5] P. Prabhakaran, P. Banerjee, "Parallel algorithm for simultaneous scheduling, binding and floorplanning in high-level synthesis," Proc. Int. Symp. on Circuits and Systems, vol. 6, pp372-376, 1998.

[6] K. Ohashi, M. Kaneko, S. Tayu, "Assignment-space exploration approach to concurrent datapath/floorplan synthesis," Proc. Int. Conf. on Computer Design, pp.370–375, 2000.

[7] D. Kim, J. Jung, S. Lee, J. Jeon, K. Choi, "Behavior-to-placed RTL synthesis with performance-driven placement," Proc. Int. Conf. on Computer Aided Design, 2001.

[8] M. Kaneko, K. Ohashi, "Assignment Constrained Scheduling under Max/Min Logic/Interconnect Delays for Placed Datapath", Proc. APCCAS2004, vol.2, pp.545-548, 2004.

[9] John P. Fishburn, "Clock Skew Optimization", IEEE Trans. on Computers, pp. 945–951, Vol.39, No. 7, 1990.

[10] 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.

[11] Bakhtiar Affendi Rosdi and Atsushi Takahashi, "Multi-clock Cycle Paths and Clock Scheduling for Reducing the Area of Pipelined Circuits," IEICE Trans. Fundamentals, Vol.E89-A, No.12, pp.3435-3442, 2006.

[12] Yuko HASHIZUME, Yasuhiro TAKASHIMA and Yuichi NAKAMURA, "Post-Silicon Clock-Timing Tuning Based on Statistical Estimation," IEICE Trans. Fundamentals, Vol.E91-A, No.9, pp.2322-2327, 2008.

[13] Xun Liu, Marios C.Papaefthymiou, Eby G. Friedman, "Retiming and Clock Scheduling for Digital Circuit Optimization", IEEE Trans. on Computer Aided Design, pp.184–203, Vol.21, No. 2, 2002.

[14] Eigo Kamibayashi, Y.Kohira and A.Takahashi "Circuit Modification Method of Semi-Synchronous Circuit," Technical report of IEICE, VLD2004-146, ICD2004-242, March 2005.

Publications

Journals

[1] Takayuki Obata and Mineo Kaneko, "Simultaneous Optimization of Skew and Control Step Assignments in RT-Datapath Synthesis," IEICE Transaction on Fundamentalsof Electronics, Communications and Computer Sciences Vol. E91-A No.12 pp. 3585-3595, Dec. 2008.

International Conferences

[2] Takayuki Obata and Mineo Kaneko, "Solvability of Simulteneous Control Step and Timing Skew Assignments in High Level Synthesis," IEEE International Symposium on Circuit and System, May 2009 (accepted).

[3] Takayuki Obata and Mineo Kaneko, "Simultaneous Optimization of Skew and Control Step Assignments in RT-Datapath Synthesis," IEEE International Symposium on Circuit and System, pp.2018-2021, May 2008.

[4] Takayuki Obata and Mineo Kaneko, "Simultaneous control-step and skew assignmentor control signals in RT-level datapath synthesis," Proc. Workshop on Synthesisand System Integration of Mixed Information Technologies, pp. 314-321, Apr. 2006.

[5] Takayuki Obata and Mineo Kaneko, "Control signal skew scheduling in RT level datapath synthesis," Proc. IEEE International Midwest Symposium on Circuits and Systems, pp. 1087-1090, Aug. 2005.

Domestic conferences/Workshops

[6] Takayuki Obata, Mineo Kaneko, "Enlarging The Solution Spaces For Schedulability Based On Skew Optimization," IEICE Technical Report (VLD2008-86) Vol.108, No.298, pp.157-162, Nov. 2008 (In Japanese).

[7] Takayuki Obata, Mineo Kaneko, "Schedulable Resouce Binding under Skew Optimization," IEICE Technical Report (VLD2008-50), Vol. 108, No.244, pp.19-24, Sep. 2008 (In Japanese).

[8] Takayuki Obata and Mineo Kaneko, "A Schedule Improvement with Skew Control in Datapath Synthesis," IEICE Technical Report (VLD2007-94), Vol. 107, No. 336, pp. 31-36, Nov. 2007.

[9] Takayuki Obata and Mineo Kaneko, “Computational complexity of simultaneous optimization of skew, schedule and clock in high-level synthesis,” IEICE Technical Report (CAS2006-75), Vol. 106, No. 512, pp. 31-36, Jan. 2007.

[10] Takayuki Obata and Mineo Kaneko, “Computational complexity of simultaneous optimization of control schedule and skew in datapath synthesis,” IEICE Techinical Report (VLD2006-65), Nov. 2006.

[11] Takayuki Obata and Mineo Kaneko, “Computational Complexity of Simultaneous Control Step and Skew Assignment in Data Path Synthesis,” IEICE 2006 Society Conference, AS-1-2, pp.”S-3”-”S-4” Sep. 2006.

[12] Takayuki Obata and Mineo Kaneko, ”Simultaneous control-step and skew assignment for control signals in RT-level datapath synthesis,” Proc. Workshop on Synthesis and System Integration of Mixed Information Technologies, pp. 314-321, Apr. 2006.

[13] Takayuki Obata and Mineo Kaneko, ”Simultaneous control-step and skew assignment for control signals in RT-level datapath synthesis,” Technical Report of IEICE (CAS2005-92), Vol. 105, No. 504, pp. 31-36, Jan. 2006.

[14] Takayuki Obata and Mineo Kaneko, ”Control signal skew scheduling for RT level datapaths,” Proc. 18th Workshop on Circuits and Systems in Karuizawa, pp. 521-526, Apr. 2005.

[15] Takayuki Obata and Mineo Kaneko, ”Control signal skew scheduling for RT level datapaths,” Technical Report of IEICE (CPSY2004-107), pp. 13-17, Mar. 2005.

[16] Takayuki Obata and Mineo Kaneko, ”Simultaneous Scheduling and Skew Assignment for Multiplexer Control in Placed Datapaths,” Technical Report of IEICE (CAS2004-74), Vol.104, No.557, pp. 13-18, Jan. 2005.