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

2007－ＳＬＤＭ－132（25）
2007／11／22

データパス合成におけるスキューを用いたスケジュール改善手法

小畑　貴之†　　金子　峰雄†

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

あらまし　論理回路を高速化する手法としてクロックスキュー及びリタイミングを併用する手法が提案されているが，RT レベル記述においては，これはレジスタやマルチプレクサの動作タイミングにスキューを導入しスケジュールを変更することに相当する．動作タイミングのスキューとスケジュールを同時に最適化する問題は NP-hard であることが証明されているが，本稿では初期スケジュールを基に，スキューの導入とスケジュールの変更により，コントロールステップ数を削減する手法を提案する．また実験によって本手法の有効性を確認する．

キーワード　高位合成，スキュー，スケジュール

A Schedule Improvement with Skew Control in Datapath Synthesis

Takayuki OBATA† and Mineo KANEKO†

† School of Information Science, Japan Advanced Institute of Science and Technology
Asahidai 1-1, Nomi, Ishikawa, 923-1211 Japan
E-mail: †{t-obata,mkaneko}@jaist.ac.jp

Abstract　As well as the schedule affects system performance, the control skew, i.e., the arrival time difference of control signals between registers, can be utilized for improving the system performance, enhancing robustness against delay variations, etc. In this paper, we discuss the simultaneous optimization of the control step assignment and the control skew assignment. Since the problem has been proved to be NP-hard, a heuristic algorithm based on critical paths in a schedule is proposed.

Key words　High-level synthesis, skew scheduling, control Scheduling

1. Introduction

In the logic level VLSI design, the clock skew is now utilized intentionally for improving system performances, enhancing the robustness against delay variations, reducing maximum peak power, etc., and significant efforts have been devoted to so-called clock-scheduling and simultaneous optimization of re-timing and clock-scheduling [1]- [3]. Recently, the importance and the impact of considering timing skew in the high level synthesis are recognized, and researches on this issue have been started at several sections [6]- [9].

High level synthesis is the transformation from the behavioral description in the algorithm level to the structural and the behavioral descriptions in the register-transfer (RT) level, and the performance of the datapath customized to a specified application has been determined mostly at this design stage. Because of the existence of several different approaches to the high level synthesis, the way of introducing an intentional skew into high level synthesis for designing higher performance VLSIs would not be unique. One possible scenario is that a conventional synthesis system incorporates intentional timing skew, and uses it to compensate mismatches of function delays. One other possible scenario is that a concurrent datapath/floorplan synthesis system [10]-[14] incorporates intentional skew, and uses it to compensate mismatches of path delays (each path delay may include function delays and signal propagation delays). Similar to the clock schedule in the logic level design, the skew-aware high level design will contribute to reducing the clock period, enhancing the robustness against delay variations. In addition, the intentional skew will also contribute to reducing the number of control steps (makespan) for a target application.

It is well-known in the logic level design that the clock skew only is not enough for achieving the highest performance, and the combination of the clock skew with the retiming technique is a promising approach. Similar to this

situation, in the skew-aware high level synthesis, the simultaneous optimization of the control step assignment and the skew assignment has a higher potential in performance optimization.

In this paper, we discuss a simultaneous optimization of the control step assignment and the skew assignment, which can be used as a common core task in various different scenarios of skew-aware high level synthesis. Taking the peculiarity of the skew assignment into consideration, we assume that resource binding and the temporal order of lifetimes of data assigned to the same register are fixed and they are specified in the input description to our problem. Our simultaneous control-step and skew optimization has been shown to be a NP-hard problem [15], whereas each of skew optimization with fixed control-step assignment and constrained control-step optimization (that is, resource binding and the temporal order of lifetimes of data assigned to the same register are fixed) with fixed skew assignment is in the class P. Major contributions of this paper are to give a heuristic algorithm for our simultaneous control-step and skew optimization problem, and to show how much the simultaneous optimization improves system performance. In the past, the intentional skew was used to shorten a clock period, and as a result, the clock period was not controlled intentionally. This paper is the first one that uses the intentional skew to shorten control steps (and hence a real application time) under a specified clock period.

This paper is organized as follows. In Section 2, we summarize basic notations with comments about our stance on scheduling issue. In Section 3, our simultaneous optimization of the control step assignment and the skew assignment is formulated. We present a heuristic algorithm in Section 4. Experimental results are shown in Section 5. Finally, we present conclusions in Section 6.

2. Background and Motivation

2.1 Structural Description of Datapath Circuit

We assume that an input algorithm to the high-level synthesis is described as a data flow graph (DFG in short) $(\mathcal{O}, \mathcal{D})$ as shown in Fig.1(a). A vertex set $\mathcal{O}$ is the set of operations and an edge set $\mathcal{D}$ indicates data dependencies between operations.

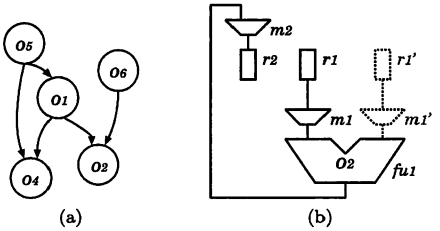


Fig. 1 Examples of DFG and RT-Level architecture.

The input algorithm is transformed to a datapath circuit by determining resource assignment, that is, a functional unit assignment $\rho : \mathcal{O} \rightarrow \mathcal{F}$ and a register assignment $\xi : \mathcal{O} \rightarrow \mathcal{R}$, where $\mathcal{F}$ is a set of functional units, $\mathcal{R}$ is a set of registers, and $\xi(o) = r$ means that the output data of an operation o is assigned to a register r (the output of o is stored in r). Interconnections and multiplexers in the datapath part is so designed that, for each operation o_j with $(o_i, o_j) \in \mathcal{D}$, the output terminal of an input register $\xi(o_i)$ is connected to the input terminal of an functional unit $\rho(o_j)$ and the output terminal of $\rho(o_j)$ is connected to the input terminal of an output register $\xi(o_j)$. A simple example of a datapath circuit is shown in Fig.1(b).

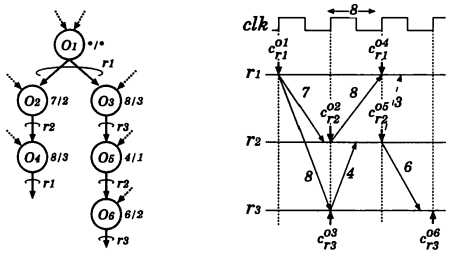
2.2 Scheduling Problem and Control Issue

Registers and multiplexers are driven by control signals. There are three kinds of control signals related to an operation $o \in \mathcal{O}$ in an input algorithm. One is control signals for two multiplexers (m1 and m1' in Fig.1(b)) located at input terminals of a FU $\rho(o)$ (fu1 in Fig.1), one for an output register $\xi(o)$ (r2 in Fig.1), and the last one for an input multiplexer (m2 in Fig.1) of the output register $\xi(o)$.

Now $\mathcal{M}$ denotes the set of all registers and multiplexers, and $\mathcal{S}$ denotes the set of all control signals. $c_x^o \in \mathcal{S}$ represents the control signal that is related to the execution of $o \in \mathcal{O}$ and is sent to $x \in \mathcal{M}$.

When the execution delay in a functional unit is dominant, wire delay is negligible, and also clock/control skew is negligible, the timing of control signals can be determined from the conventional operation schedule without ambiguity. That is, it is enough to determine the start control step and the end control step for each operation with considering maximum execution delay only. However, it is not the case for LSIs with non-negligible wire delay and clock/control skew, and we will now consider control signals, instead of operations, as the objects to be scheduled [14]. (Please note that, if we set parameters appropriately, the control signal schedule can simulate the operation schedule. In this sense, the control signal schedule is a generic concept including the operation schedule.) It has an additional merit that, considering the timing of control signals directly, the so-called wave pipelining, which utilizes not only maximum path delay information but also minimum path delay information, can be naturally embodied.

The behavior of the datapath circuit is determined by the arrival timing of control signals to registers and multiplexers. The arrival timing is partly determined by the control step assignment, and the rest by the timing skew. 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 $\sigma : \mathcal{S} \rightarrow Z_+$ as a con-



(a) DG (b) Design 1

(c) Design 2 (d) Design 3

Fig. 2 Necessity of skew aware scheduling.

trol schedule. $\tau(x)$ for $x \in \mathcal{M}$ is the skew value assigned to x. In total, the control signal c^o_x reaches x at the time $\sigma(c^o_x) \cdot clk + \tau(x)$, where clk is a clock period.

2.3 Motivational Example

Fig.2(a) shows a schedule of 6 control signals. The number written beside a slant solid (broken) arrow shows maximum (minimum) path delay of the data propagation. The schedule requires 3 clock cycles. This is an optimal schedule under zero skew if the number of control steps is restricted to smaller than or equal to 3, and its minimum clock period is 8 (the total computation time is $8 \times 3 = 24$). When we assign skew $(\tau(r1), \tau(r2), \tau(r3)) = (0, -0.5, 0.5)$, the minimum clock period can be reduced to 7.5 (the total computation time is now $7.5 \times 3 + 0.5 = 23$). The situation is illustrated in Fig.2(b). Fig.2(c) shows an optimal schedule and skew assignment. The minimum clock period is now 5 (the total computation time is now, $5 \times 3 + 3 = 18$).

This example shows that we can not obtain an optimal solution by applying the skew assignment and the control step assignment separately, and so we should schedule control signals considering skew.

3. Simultaneous Optimization of Control Step and Skew Assignments

3.1 Formulation of the Problem

Our simultaneous optimization problem, (σ,τ,clk)-optimization, receives (1) data flow graph G, (2) resource assignment ρ, ξ, and (3) the execution order of operations assigned to the same FU and the production order

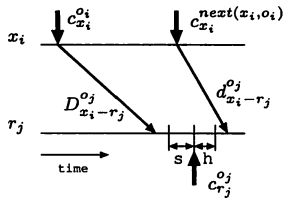


Fig. 3 Setup / Hold constants.

of data assigned to the same register (a function $next$ reflects the corresponding information) as the input instance, and outputs σ, τ and clk.

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

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

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

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

On the other hand, the arrival of $c^{o_j}_{r_j}$ has to be earlier than the destruction of the result of o_j. This is called "hold constraint" and is formulated as

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

where $d^{o_j}_{x_i - r_j}$ is the minimum path delay from x_i to r_j related to the execution of o_j, and h is the hold time of r_j. $next(x_i, o_i)$ is the operation next to o_i on the resource x_i.

It is natural and convenient to constrain $0 \leqq \tau(x) < clk$ for all $x \in \mathcal{M}$. It is clear that such restriction does not cause the loss of optimality.

In general, the objective of the scheduling is the minimization of the computation time and the size of a resultant circuit. When the binding ξ and ρ are fixed, the minimization of the size of a circuit is reduced to the minimization of 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 required control steps). Since $|\mathcal{M}|$ is fixed, CS is our objective to be minimized. On the other hand, the computation time of a circuit can be evaluated with $clk \cdot CS$. Finally, we choose

$$clk \cdot CS + \lambda \cdot CS$$

as the objective to be minimized, where λ is a weighting parameter similar to what we can see in many multi-purpose optimization problems.

3.2 Partial Problems

Depending on a design strategy, design methodology, target application, design constraints, etc., we may encounter several partial problems.

(τ,clk)-optimization is the problem to optimize τ and clk while keeping control schedule σ unchanged. Because τ and clk take real numbers, (τ,clk)-optimization problem can be formulated as LP problem. An efficient graph theoretic approach has been proposed [6].

(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 can be considered also as a candidate subroutine for solving the original (σ,τ,clk)-optimization, that is, for example repeating (σ,τ)-optimization with a systematic sweep of clk. NP-hardness of (σ,τ)-optimization has been shown in [15] as well as NP-hardness of (σ,τ,clk)-optimization problem [16]. In the rest of this paper, we investigate solution algorithm for (σ,τ)-optimization problem.

4. Heuristic Algorithm

By our preliminary study, we have proven that, even if the execution sequence of operations assigned to the same resource is fixed and only the control step assignment remains unfixed, (σ, τ)-optimization under a fixed clock period is $\mathcal{NP}$-hard [15]. In this section, we show a heuristic algorithm for this (σ,τ)-optimization problem.

4.1 Skew Constraint Graph

From (1) and (2), we have

$$\tau_r - \tau_m \geqq (\sigma_m^{o_p} - \sigma_r^{o_k}) \cdot clk + \tau_{err} + D_{m-r}^{o_k} + s \quad (3)$$

$$\tau_m - \tau_r \geqq (\sigma_r^{o_k} - \sigma_m^{next(m,o_p)}) \cdot clk + \tau_{err} - d_{m-r}^{o_k} + h \quad (4)$$

We generate a skew constraint multigraph $G_\sigma = (V, E)$ from (3) and (4) as shown in Fig.4. V is a set of multiplexers, registers and one auxiliary source node v_s. A set of weighted edges E is the union of a set of edges reflecting (3) or (4) (i.e., an edge (m, r) with weight $(\sigma_m^{o_p} - \sigma_r^{o_k}) \cdot clk + \tau_{err} + D_{m-r}^{o_k} + s$

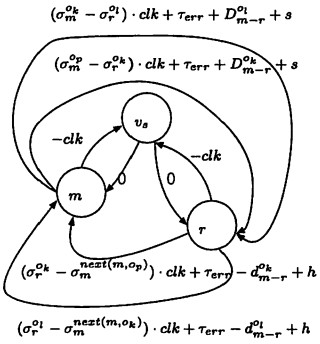


Fig. 4 Skew constraint multigraph

or an edge (r, m) with weight $(\sigma_r^{o_k} - \sigma_m^{next(m,o_p)}) \cdot clk + \tau_{err} - d_{m-r}^{o_k} + h$ over all operations), and a set of auxiliary edges $\{(m, v_s) | m \in V \setminus v_s\} \cup \{(v_s, m) | m \in V \setminus v_s\}$. Edge weights for $\{(m, v_s) | m \in V \setminus v_s\}$ and $\{(v_s, m) | m \in V \setminus v_s\}$ are $-clk$ and 0, respectively. Then, skew assignment problem is now considered as the problem to assign real values to vertices in G_σ, and maximum path lengths from v_s to other vertices give us a solution, i.e., skews of registers and multiplexers. If G_σ has a positive cycle, feasible skew schedule does not exist.

4.2 Schedule Constraint Graph

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

$$\sigma(c_{r_j}^{o_j}) - \sigma(c_{x_i}^{o_i}) \geqq \left\lceil \left(\tau(x_i) - \tau(r_j) + t_{err} + D_{x_i - r_j}^{o_j} + s \right) / clk \right\rceil, \quad (5)$$

$$\sigma(c_{x_i}^{next(x_i,o_i)}) - \sigma(c_{r_j}^{o_j}) \geqq \left\lceil \left(\tau(r_j) - \tau(x_i) + t_{err} - d_{x_i - r_j}^{o_j} + h \right) / clk \right\rceil \quad (6)$$

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) or (6), 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. A path which gives CS is called a critical path. If G_τ has a positive cycle, feasible schedule does not exist.

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

Suppose we have computed τ from G_σ, 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 $(\tau(r_j) - \tau(x_i))$ mod clk is equal to either "$(t_{err} + D_{x_i - r_j}^{o_j} + s)$ mod clk" or "$(t_{err} - d_{x_i - r_j}^{o_j} + h)$ mod clk" depending on the edge weight. Therefore, we can consider the skew optimization problem as the problem to extract a spanning tree

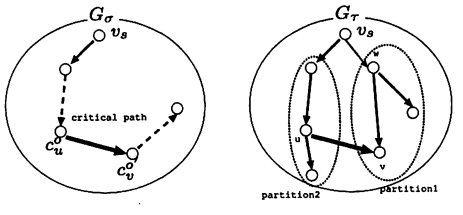


Fig. 5 We add an edge on a critical path in G_σ to T.

Step1. Generate G_τ and G_σ

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

Step3. Compute τ from T. Compute σ from G_σ. Let $\mathcal{P}$ be a critical path in G_σ.

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 $\tau_{(u,v)}$ and the number of control steps $CS_{(u,v)}$.

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

Fig. 6 Heuristic algorithm

from G_σ.

Since the right hand side of (5)-(6) has a ceiling, if the relative skew $(\tau(r_j) - \tau(x_i))$ mod clk is equal to $(t_{err} + D^{o_j}_{x_i - r_j} + s)$ mod clk or $(t_{err} - d^{o_j}_{x_i - r_j} + h)$ mod clk, the weight of the edge reflecting inequality (5) or (6) is minimized. Therefore, to minimize CS, it is efficient to set relative skew to $(t_{err} + D^{o_j}_{x_i - r_j} + s)$ mod clk or $(t_{err} - d^{o_j}_{x_i - r_j} + h)$ mod clk for as many edges as possible.

Our heuristic algorithm is shown in Fig.6. We start with the spanning tree T whose edge set is $\{(v_s, m) | m \in V \setminus v_s\}$ i.e. $\tau(m) = 0$ for all m. We replace (v_s, m) by an edge corresponding to an edge on a critical path in G_σ one with one. We use a partitioning to know which edge we can add and which edge we have to remove in order to keep T a tree. 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 Figure 5 shows the replacement of edges. We add (u,v) to T only if u and v belong to different partite sets.

5. Experiments

The proposed algorithm 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.

Path 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. The minimum register-multiplexer and FU-multiplexer delays are chosen from numbers from 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 1.1 to 1.4 times larger than its minimum delay.

Table 1 Experimental results

Algorithm	#fu	#reg	clk	CS		time(ms)	
				n/s	w/s	n/s	w/s
Jaumann	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
	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
Lattice	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
	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
Elliptic	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
	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

We have prepared 2 input instances for each input algorithm, each instance has different resource assignment, different delay assignment and different operation order. For each instance, we have applied schedule optimization without skew optimization (assuming zero skew) and proposed algorithm.

Table 1 shows some of experimental results. The column "#fu","#reg","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 7 through 9 plot the application time (i.e., $CS \times clk$) vs. clock period. Those plots are obtained by applying our algorithm repeatedly with increasing clk by 1 at a time. A thin doted line represents the lower bound of $CS \times clk$.

Experimental results show the effectiveness of our proposed algorithm, and also the potential of the simultaneous optimization of skew and control step assignments in improving system performance.

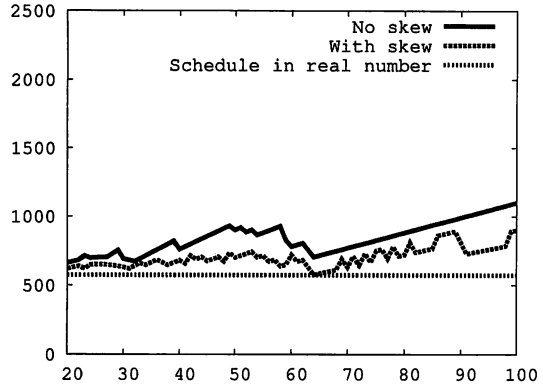


Fig. 7 Application time ($CS \times clk$) vs. clk for Jaumann

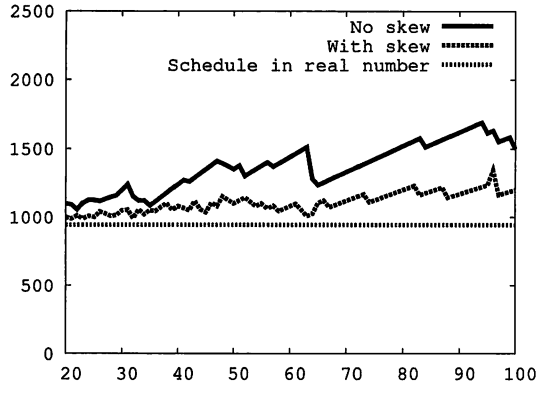


Fig. 8 Application time ($CS \times clk$) vs. clk for Lattice

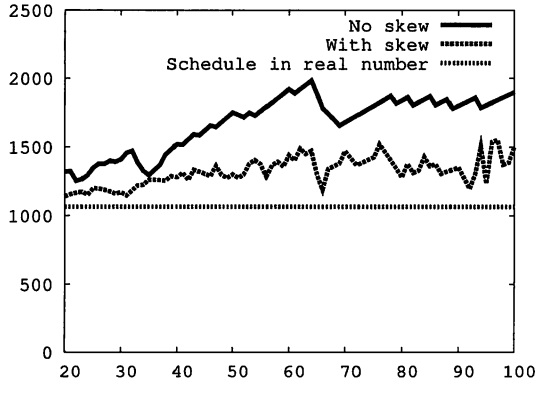


Fig. 9 Application time ($CS \times clk$) vs. clk for Elliptic

6. Conclusion

We have introduced a novel optimization, simultaneous schedule (control step assignment) and skew optimization problem, and we have proposed a heuristic algorithm for the simultaneous control step and skew optimization under a given clock period. The algorithm has the potential to play a central role in various scenarios of the skew-aware high level synthesis. Relation between the simultaneous optimization of skew and re-timing in the logic level and our problem in the register-transfer level is one of the interesting future works.

References

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

[2] R. B. Deokar and S. S. Sapatnekar, "A graphtheoretic approach to clock skew optimization," Proc. IEEE Int. Symp. Circuits and Systems, pp. 1.407–.410, 1994.

[3] 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.

[4] E. Kamibayashi, Y. Kohira and A. Takahashi, "Circuit Modification Method of Semi-Synchronous Circuit," Technical report of IEICE, VLD2004–46, ICD2004-242, March 2005.

[5] Y. Kohira, A. Takahashi, "Clock Period Minimization Method of Semi-Synchronous Circuits by Delay Insertion", IEICE Trans. Fundamentals, Vol.E88-A No.4, pp.892-898, 2005.

[6] Takayuki Obata and Mineo Kaneko "Control Signal Skew Scheduling for RT Level Datapaths," Proc. of IEICE 18th Karuizawa Workshop on Circuits and Systems, pp. 521-526, April 2005.

[7] Takayuki Obata, Mineo Kaneko, "Control Signal Skew Scheduling in RT Level Datapath Synthesis", Proc. of IEEE International Midwest Symposium on Circuits and Systems, CD–ROM ISBN:0-7803-9198-5, August 2005.

[8] Takayuki Obata, Mineo Kaneko, "Simultaneous Control-step and Skew Assignment for Control Signals in RT-Level Datapath Synthesis", Proceedings of SASIMI2006, pp.314–321, April 2006.

[9] Shih-Hsu Huang, Chun-Hua Cheng, Yow-Tyng Nieh, Wei-Chieh Yu, "Register binding for clock period minimization", Proc. of the 43rd annual Conference on Design Automation, pp.439-444, July 2006.

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

[11] 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.

[12] 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.

[13] 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.

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

[15] Takayuki Obata, Mineo Kaneko, "Computational Complexity of Simultaneous Optimization of Control Schedule and Skew in Datapath Synthesis", IEICE technical report, VLD2006-65, DC2006-52 pp. 83–88, November 2006.

[16] Takayuki Obata, Mineo Kaneko, "Computational Complexity of Simultaneous Optimization of Skew, Schedule and Clock in High-Level Synthesis", IEICE technical report, CAS2006-75, pp. 31–36, January 2007.