Concurrent Skew and Control Step Assignments in RT-Level Datapath Synthesis

Takayuki Obata Mineo Kaneko
School of Information Science, Japan Advanced Institute of Science and Technology
Nomi-shi, Ishikawa 923-1292 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. The simultaneous optimization of the control step assignment and the control skew assignment is more powerful technique in improving performance. By our preliminary study, we have proven that, even if the execution sequence of operations assigned to the same resource is fixed, the simultaneous optimization problem under a fixed clock period is $\mathcal{NP}$-hard. In this paper, we propose a heuristic algorithm for the simultaneous control step and skew optimization under given clock period, and we show how much the simultaneous optimization improves system performance. This paper is the first one that uses the intentional skew to shorten control steps (and hence a real application time) under a specified clock period. The proposed algorithm has the potential to play a central role in various scenarios of skew-aware high level synthesis.**

I. 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]-[4]. Recently, the importance and the impact of considering timing skew in the high level synthesis are recognized, and related researches have been started at several sections[5]-[6].

Because of the existence of several different approaches to the high level synthesis, the way of introducing intentional skew (intentionally controlled timing difference between the beginning of a control step and the working timing of each register and multiplexer) into high level synthesis for designing higher performance VLSIs is not 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 [7]-[9] 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. It is interesting that the intentional skew will also contribute to reducing the number of control steps (makespan) for a target application.

It is well-known in the logic level design that the clock skew only is not enough for the highest performance, and the combination of the clock skew with the re-timing technique is a promising approach. Similar to this situation, in the skew-aware high level synthesis, the simultaneous optimization of the control step assignment and the skew assignment has a higher potential in performance optimization. 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 they are given in the input description to our problem, and we try to optimize skew and control step assignments under those constraints. It is clear that, if we have such a tool to solve this problem, it can be used as a sub-tool for optimizing resource binding and temporal order of lifetimes.

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.

II. Back Ground and Motivation

A. Structural and Behavioral Descriptions of Datapath Circuit

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

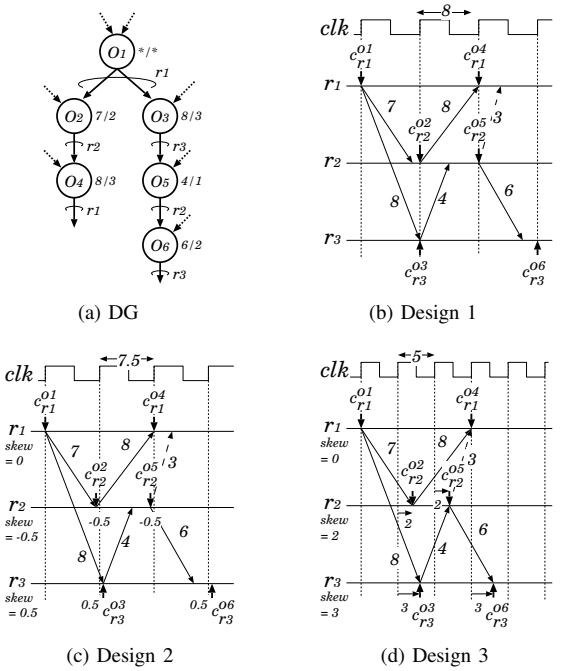
The input algorithm is transformed to the datapath circuit by determining resource assignment, that is, the functional unit assignment $\rho : \mathcal{O} \rightarrow \mathcal{F}$ and the register assignment $\xi : \mathcal{O} \backslash \mathcal{U} \rightarrow \mathcal{R}$, where $\mathcal{F}$ is a set of functional units, $\mathcal{R}$ is a set of registers, $\mathcal{U}$ is a set of operations whose outputs are not written to registers, and $\xi(o) = r$ means that the output data of the operation o is assigned to the register r (the output of o is written in r).

We let $\mathcal{M}$ be the set of all registers (and multiplexers), and $\mathcal{S}$ denotes the set of all control signals, where $c_x^o \in \mathcal{S}$ represents the control signal which is related to the execution of $o \in \mathcal{O}$ and is sent to $x \in \mathcal{M}$. The arrival timing is partly determined by the control step assignment, 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 control schedule. $\tau(x)$ for $x \in \mathcal{M}$ is the skew value assigned to x. 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 the clock period.

B. Motivational Example

Fig.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 multiplexers if necessary, and 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 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.1(a), the maximum and the minimum delays are indicated like

978-1-4244-1684-4/08/$25.00 ©2008 IEEE



(a) DG (b) Design 1

(c) Design 2 (d) Design 3

Fig. 1. Necessity of skew aware scheduling.

7/2 for O_2, 8/3 for O_3, etc. The assignment of data to registers is indicated with r_i beside each arc.

Fig.1(b) shows a schedule of 6 control signals $c^{o_1}_{r_1}$, $c^{o_2}_{r_2}$, $c^{o_3}_{r_3}$, $c^{o_4}_{r_1}$, $c^{o_5}_{r_2}$, and $c^{o_6}_{r_3}$. The number written beside a slant solid (broken) arrow shows maximum (minimum) delay of the corresponding operation. The schedule requires 4 control steps, and its minimum clock period is 8 (the total computation time is $8 \times 4 = 32$). This is an optimal schedule under zero skew if the number of control steps is restricted to smaller than or equal to 4. 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 $7.5 \times 4 = 30$). The situation is illustrated in Fig.1(c). Fig.1(d) shows an optimal schedule and skew assignment. If we try to keep the number of control steps to 4, the minimum clock period is now 5 (totally, $5 \times 4 = 20$).

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

III. Simultaneous Optimization of Control Step and Skew Assignments

A. Formulation of the Problem

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

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 $c^{o_j}_{r_j}$ has to be later than the arrival of the result of o_j) is formulated as

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

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

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

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

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

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.

B. Partial Problems

Depending on the 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. Because τ and clk take real values, (τ,clk)-optimization problem can be formulated as LP problem.

(σ,clk)-optimization is the problem to optimize σ and clk under given skew τ. Conventional high-level synthesis systems treated (σ,clk)-optimization under 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. In the rest of this paper, we investigate (σ,τ)-optimization problem.

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

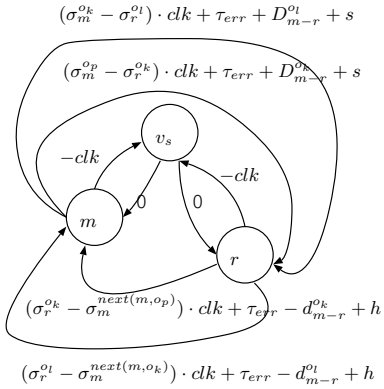


Fig. 2. Skew constraint multigraph

A. Skew Constraint Graph

From (1)-(2), we have

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

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

We generate a skew constraint multigraph $G_\tau = (V, E)$ from (3-4) as shown in Fig.2. 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 $(\tau_m^{o_p} - \tau_r^{o_k}) \cdot clk + \tau_{err} + D_{m-r}^{o_k} + s$ 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 $\{(m, v_s)|m \in V \setminus v_s\} \cup \{(v_s, m)|m \in V \setminus v_s\}$. 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_τ, and maximum path lengths from v_s to other vertices gives us a solution, i.e., skew of registers and multiplexers. If G_τ has a positive cycle, feasible skew schedule does not exist.

B. Schedule Constraint Graph

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

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

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

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.

C. Heuristic Algorithm for (σ, τ)-optimization Problem

Suppose we have computed τ from G_τ, and consider the union T of a longest path from v_s to each node. Then, T is a spanning tree, and for each edge (x_i, r_j) in T, relative skew $(\sigma(r_j) - \sigma(x_i)) \bmod clk$ is equal to either "$(t_{err} + D_{x_i - r_j}^{o_j} + s) \bmod clk$" or "$(t_{err} - d_{x_i - r_j}^{o_j} + h) \bmod clk$" depending on the edge weight. Therefore, we can consider the skew optimization problem as the problem to extract a spanning tree from G_τ.

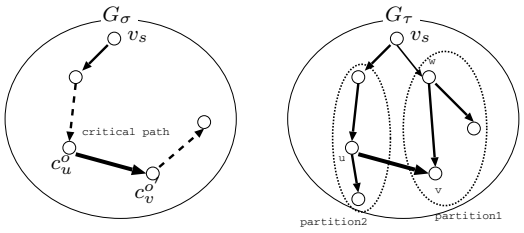


Fig. 3. We add an edge on a critical path in G_σ to T.

Step1. Generate G_τ and G_σ
Step2. Generate an initial solution $T \subset G_\tau$.
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. 4. Heuristic algorithm

Since the right hand side of (5)-(6) has a ceiling, if the relative skew $(\sigma(r_j) - \sigma(x_i)) \bmod clk$ is equal to $(t_{err} + D_{x_i - r_j}^{o_j} + s) \bmod clk$ or $(t_{err} - d_{x_i - r_j}^{o_j} + h) \bmod clk$, the weight of the edge reflecting inequality (5) or (6) is minimized. Therefore, to minimize CS, it is efficient to set relative skew to $(t_{err} + D_{x_i - r_j}^{o_j} + s) \bmod clk$ or $(t_{err} - d_{x_i - r_j}^{o_j} + h) \bmod clk$ for as many edges as possible in a critical path i.e. 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 figure 4. We start with the spanning tree T in G_τ whose edge set is $\{(v_s, m)|m \in V \setminus v_s\}$ i.e. $\sigma(m) = 0$ for all m. We replace (v_s, m) by an edge corresponding to an edge on a critical path in G_σ one by 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 component in $T \setminus \{(v_s, x)|x \in V\}$ 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 component to which v belongs to. G_τ in Figure 3 shows the replacement of edges. We add (u, v) to T only if u and v belong to different partite sets.

V. Experiments

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 3-25 and the minimum multiplexer-register and multiplexer-FU delays are chosen from 2-15. The maximum delay of each path is 1.1-1.4 times larger than its minimum delay.

TABLE I
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
			80	14	11	0.123	11.9
	7	7	20	33	31	0.114	1.72
			40	19	17	0.114	3.05
			80	11	9	0.115	10.5
Lattice	3	5	20	55	50	0.075	2.12
			40	31	27	0.076	3.05
			80	19	15	0.075	3.80
	4	5	20	50	46	0.078	2.76
			40	29	25	0.078	3.33
			80	17	14	0.077	5.02
Elliptic	8	13	20	66	57	0.241	42.1
			40	38	33	0.241	74.0
			80	23	16	0.239	96.5
	8	14	20	67	58	0.245	42.2
			40	38	31	0.244	51.6
			80	22	18	0.243	101

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 I shows some of experimental results. The column "#fu","#reg","clk" represent the number of function units, registers, and clock period of each instance, respectively. The column "CS" represents the number of control steps (makespan) of output schedule with proposed algorithm (the subcolumn "n/s") and zero skew (the subcolumn "w/s"). The column "time" represents the computation time in milli-seconds. Figures 5-7 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 $CS \times clk$.

For almost all input instances, we have better results than zero skew control step assignment. 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.

VI. CONCLUSIONS

We have introduced a novel optimization problem, 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 given clock period. The algorithm has the potential to play a central role in various scenarios of skew-aware high level synthesis. Relation between the simultaneous optimization of skew and re-timing in logic level and our problem in high 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. 1.407-1.410, 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.

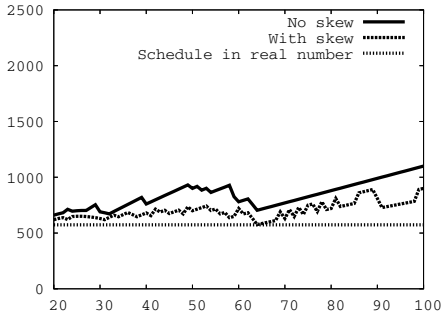


Fig. 5. Application time ($CS \times clk$) vs. clk for Jaumann

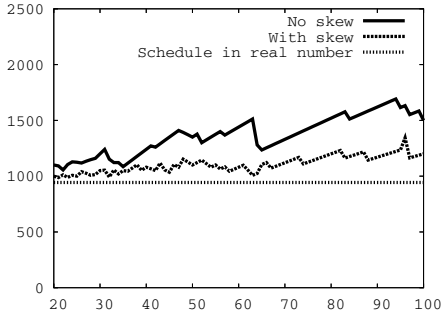


Fig. 6. Application time ($CS \times clk$) vs. clk for Lattice

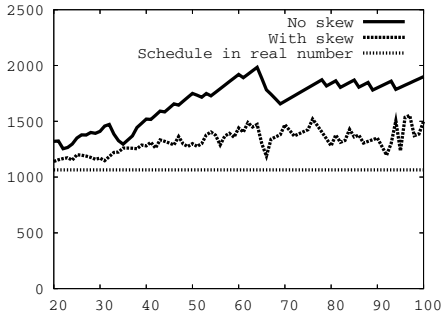


Fig. 7. Application time ($CS \times clk$) vs. clk for Elliptic

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.