Register Binding for Clock Period Minimization

Shih-Hsu Huang, Chun-Hua Cheng, Yow-Tyng Nieh, Wei-Chieh Yu
Department of Electronic Engineering, Chung Yuan Christian University,
Chung Li, Taiwan, R.O.C.
{shhuang, g9402601, g9202601, g9376042}@cycu.edu.tw

ABSTRACT

In modern high-speed circuit design, the clock skew has been widely utilized as a manageable resource to improve the circuit performance. However, in high-level synthesis stage, the circuit is never optimized for the utilization of clock skew. This paper is the first attempt to the high-level synthesis of non-zero clock skew circuits. First, we show that the register binding in high-level synthesis stage has a significant impact on the clocking constraints between registers. As a result, different register binding solutions lead to different smallest feasible clock periods. Then, based on that observation, we formulate the problem of register binding for clock period minimization. Given a constraint on the number of registers, our objective is to find a minimum-period register binding solution. Experimental data show that, in most benchmark circuits, the lower bound of the clock period can be achieved without any extra overhead on the number of registers.

Categories and Subject Descriptors

B.5.2 [Register-Transfer-Level Implementation]: Design Aids – *Automatic Synthesis*, *Optimization*.

General Terms

Algorithms, Design, Performance.

Keywords

High-Level Synthesis, Clock Skew, Timing Optimization.

1. INTRODUCTION

In logic synthesis stage, the clock skew has been widely utilized as a manageable resource for clock period minimization [1]. Several graph-based algorithms [2,3] have been proposed to solve the optimal clock skew scheduling problem in polynomial time complexity. However, no attention has been paid to the high-level synthesis of non-zero clock skew circuits. Conventional high-level synthesis assumes that the clock skew is zero; as a result, the circuit is never optimized for the utilization of clock skew.

In fact, the register binding [4-7] in high-level synthesis stage has a significant impact on the clocking constraints between registers. Different register binding solutions lead to different smallest feasible clock periods. Intuitively, if there is no register sharing (i.e., each register represents only one variable), the lower bound of the clock period can be achieved. However, this register binding solution (i.e., no register sharing) is impractical. This paper studies the register binding for clock period minimization.

Our work is the first attempt to the high-level synthesis of non-zero clock skew circuits. Given a scheduled data flow graph (DFG) and a constraint on the number of registers, we minimize the clock period via the application of simultaneous register binding and optimal clock skew scheduling. First, we use mixed integer linear programming (MILP) to formulate the problem. Note that our MILP formulation guarantees obtaining the optimal solution. We also propose a heuristic algorithm to obtain a near-optimal solution. The time complexity of our heuristic algorithm is $O(m*n*|T_{OCSS}|)$, where m is the number of variables in the scheduled DFG, n is the number of registers, and $|T_{OCSS}|$ is the time complexity of optimal clock skew scheduling used.

2. REGISTER BINDING

A behavioral description can be represented by a DFG, where each vertex corresponds to an operation, and each directed edge corresponds to data dependency or control relation. A scheduled DFG is a DFG in which each operation is scheduled into a proper control step to start its execution. Note that, in the final implementation, each control step corresponds to a clock cycle, each operation is implemented by a combinational logic (a functional unit), and each variable is stored in a register.

Let's consider the scheduled DFG shown in Figure 1(a). This scheduled DFG has 4 control steps, 8 operations, and 7 variables. For example, operation o_5 is an addition operation and performed at control step 2. Note that the addition operation is implemented by a combinational logic, e.g., an adder. The inputs of operation o_5 are variable a and variable b, the output of operation o_5 is variable e.

The lifetime of a variable is defined as the time interval from its definition to its last use. Two variables can share the same register, if and only if they do not have overlapping lifetimes. Using the scheduled DFG shown in Figure 1(a) as an example, Figure 1(b) gives the lifetime of each variable. For example, the lifetime of variable a is control step 2, and the lifetime of variable d is control step 3. Therefore, variable a and variable d can share the same register.

The *register binding* problem is to assign the variables in the scheduled DFG to the registers. Conventionally, the objective of register binding is to minimize the number of registers. The widely used register binding algorithms include left edge algorithm [4] and clique partitioning approach [5].

Suppose that the left edge algorithm is applied to the scheduled DFG shown in Figure 1(b). We find that the minimum number of registers is 3. Variable c and variable f are assigned to register R_1, variable a, variable d, and variable g are assigned to register R_2, and variable b and variable e are assigned to register R_3. In this paper, we describe the register binding solution in the following form: $R_1 = \{c,f\}$, $R_2 = \{a,d,g\}$, and $R_3 = \{b,e\}$.

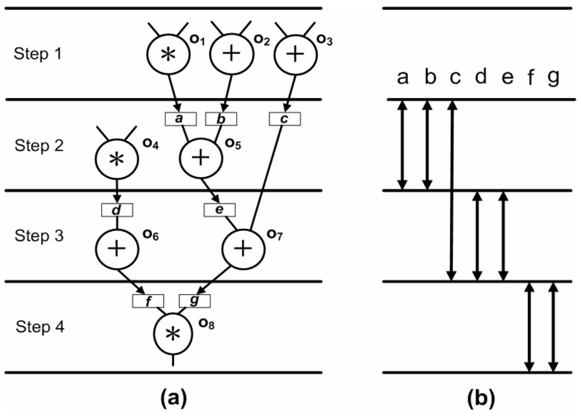

Figure 1: (a) Scheduled DFG. (b) Lifetime analysis.

3. CLOCK SKEW OPTIMIZATION

In this Section, we introduce the *circuit graph* and the *constraint graph* to model the optimal clock skew scheduling problem in high-level synthesis stage. Note that, even if the scheduled DFG has chained operations and multi-cycle operations, the optimal clock skew scheduling technique is still applicable. For the convenience of presentation, here, we do not consider chained operations and multi-cycle operations. However, the extensions to chained operations and multi-cycle operations are straightforward.

A data path from register R_i to register R_j is defined as the combinational logic from register R_i to register R_j. Suppose that the input variable of operation o_k is assigned to register R_i, and the output variable of operation o_k is assigned to register R_j. Then, the data path from register R_i to register R_j performs operation o_k. Note that the same data path may perform different operations at different control steps. Therefore, the minimum delay (maximum delay) of a data path is the minimum delay (maximum delay) among all the operations performed in the data path.

Given a scheduled DFG and a register binding solution, we can model the edge-triggered circuit as a *circuit graph*. In the circuit graph, each vertex represents a register and each directed edge represents a data path. A special vertex called the *host* is introduced for the synchronization with primary inputs and primary outputs. Each directed edge $R_i \rightarrow R_j$ is associated with a weight $(\min(R_i,R_j), \max(R_i,R_j))$, where $\min(R_i,R_j)$ and $\max(R_i,R_j)$ are the minimum delay and the maximum delay of the data path from register R_i to register R_j, respectively.

Let T_i denote the clock arrival time of register R_i. Note that T_i is defined as the time relative to a global time reference. Thus, T_i may be a negative value. For a data path from register R_i to register R_j, there are two types of clocking constraints: setup constraint and hold constraint. To prevent the data reaching a register too late relative to the following clock pulse, the clock skew must satisfy the following setup constraint: $T_i - T_j \leq P - \max(R_i,R_j)$, where P is the clock period. To prevent the same clock pulse triggering the same data into two adjacent registers, the clock skew must satisfy the following hold constraint: $T_j - T_i \leq \min(R_i,R_j)$. Both the two inequalities define a permissible clock skew range of a data path. We say that a circuit graph (or a register binding solution) works with the clock period P, if and only if there is a clock skew schedule (i.e., a solution of clock arrival times of registers) that satisfy all the clocking constraints.

The *optimal clock skew scheduling* problem [1] is to determine the smallest feasible clock period of a circuit graph and find the corresponding clock skew schedule for the circuit graph to work with the smallest feasible clock period. Several graph-based algorithms [2,3] have been proposed to solve the optimal clock skew scheduling problem in polynomial time complexity. In general, the clocking constraints of a circuit graph G is modeled by a *constraint graph* $G_{cg}(G)$, where each vertex represents a register and each directed edge represents a constraint. Note that each directed edge $R_i \rightarrow R_j$ associated with a weight $w_{i,j}$ corresponds to the constraint $T_j - T_i \leq w_{i,j}$. Therefore, each data path from register R_i to register R_j in the circuit graph G has the following two directed edges in the constraint graph $G_{cg}(G)$: the setup constraint is modeled as a directed edge $R_j \rightarrow R_i$ associated with a weight $w_{j,i} = P - \max(R_i,R_j)$, and the hold constraint is modeled as a directed edge $R_i \rightarrow R_j$ associated with a weight $w_{i,j} = \min(R_i,R_j)$. From [2], we have the following lemma.

Lemma 1: Let G denote a circuit graph. If the clock period *c* is the smallest feasible clock period of circuit graph G, then the constraint graph $G_{cg}(G)$ contains at least one critical cycle (i.e., the summation of weights is zero) when the clock period is *c*.

In the following, we use the scheduled DFG shown in Figure 1(a) for illustration. Suppose that the maximum delay and the minimum delay of a multiplier (multiplication operation) are 16 and 12, respectively, and the maximum delay and the minimum delay of an adder (addition operation) are 4 and 3, respectively. Therefore, the longest combinational delay is 16. If the clock skew is zero, the clock period is at least 16.

Assume that the register binding solution is $R_1 = \{a\}$, $R_2 = \{b\}$, $R_3 = \{c\}$, $R_4 = \{d\}$, $R_5 = \{e\}$, $R_6 = \{f\}$, and $R_7 = \{g\}$. Figure 2(a) gives the corresponding circuit graph G1. We use the directed edge $R_1 \rightarrow R_5$ as an example to illustrate the calculation of the weight. Variable a is the input variable of operation o_5 and assigned to register R_1, and variable e is the output variable of operation o_5 and assigned to register R_5. Therefore, the data path from register R_1 to register R_5 performs operation o_5 (at control step 2). Note that operation o_5 is an addition operation, and its minimum delay and maximum delay are 3 and 4, respectively. Moreover, in fact, the data path from register R_1 to register R_5 does not perform any other operation except for operation o_5. Therefore, the directed edge $R_1 \rightarrow R_5$ in the circuit graph G1 is associated with a weight $(\min(R_1,R_5), \max(R_1,R_5)) = (3,4)$.

For the circuit graph G1, Figure 2(b) gives the corresponding constraint graph $G_{cg}(G1)$, in which the setup constraint is drawn in solid line and the hold constraint is drawn in dotted line. Using the data path from register R_1 to register R_5 as an example, the setup constraint is modeled as a directed edge $R_5 \rightarrow R_1$ and associated with a weight P-4 (i.e., $T_1 - T_5 \leq P - 4$), and the hold constraint is modeled as a directed edge $R_1 \rightarrow R_5$ and associated with a weight 3 (i.e., $T_5 - T_1 \leq 3$). After applying the optimal clock skew scheduling, we find that the smallest feasible clock period is 12. Note that, when the clock period is 12, a critical cycle host$\rightarrow R_6 \rightarrow R_4 \rightarrow$host exists in the constraint graph $G_{cg}(G1)$. The optimal clock skew schedule is $T_{host} = 0$, $T_1 = 12$, $T_2 = 3$, $T_3 = 3$, $T_4 = 4$, $T_5 = 4$, $T_6 = -4$, and $T_7 = -4$. Compared with the zero clock skew circuit, the optimal clock skew scheduling reduces the smallest feasible clock period from 16 to 12.

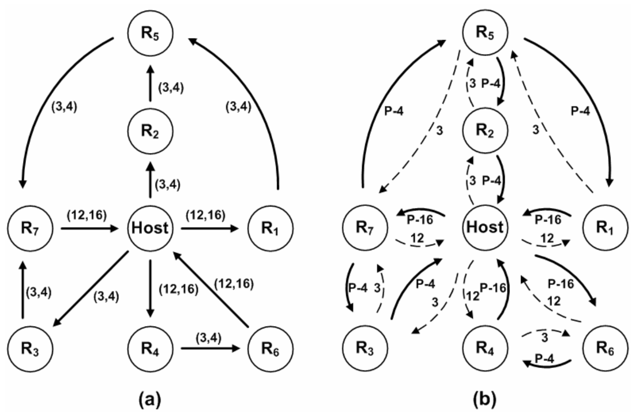
Figure 2: (a) Circuit graph G1. (b) Constraint graph $G_{cg}(G1)$.

4. MOTIVATIONS

Previous register binding algorithms [4-7] do not take the utilization of clock skew into account. However, in fact, different register binding solutions impose different clocking constraints between registers. As a result, different register binding solutions lead to different smallest feasible clock periods (achieved by the optimal clock skew scheduling). In the following, we use the scheduled DFG shown in Figure 1(a) to illustrate our observations.

First, we discuss the lower bound of the clock period that the application of simultaneous register binding and optimal clock skew scheduling can achieve. If there is no register sharing (i.e., each register represents only one variable), the register binding solution works with this lower bound. However, since there is no register sharing, this register binding solution uses too many registers and thus it is impractical. Let's consider the register binding solution in which $R_1 = \{a\}$, $R_2 = \{b\}$, $R_3 = \{c\}$, $R_4 = \{d\}$, $R_5 = \{e\}$, $R_6 = \{f\}$, and $R_7 = \{g\}$. As studied in Section 3, the smallest feasible clock period of this register binding solution achieves 12, which is the lower bound of the clock period. However, this register binding solution uses 7 registers.

On the other hand, although we can use the left edge algorithm [4] to minimize the number of registers, the register binding solution obtained by left edge algorithm often restricts the utilization of clock skew. Let's consider the register binding solution obtained by the left edge algorithm: $R_1 = \{c,f\}$, $R_2 = \{a,d,g\}$, and $R_3 = \{b,e\}$. Figure 3(a) gives the corresponding circuit graph G2. Then, for the circuit graph G2, we derive the corresponding constraint graph $G_{cg}(G2)$ shown in Figure 3(b). We find that, when the clock period is 16, a critical cycle host→R_2→host already exists in the constraint graph $G_{cg}(G2)$. Therefore, even if the optimal clock skew scheduling is applied, the smallest feasible clock period is still 16. In other words, in this example, if we adopt the register binding solution obtained by left edge algorithm, the clock skew cannot be utilized to further improve the circuit performance.

In fact, in this example, there is a register binding solution that not only uses the minimum number of registers but also works with the lower bound of the clock period. Let's consider the following register binding solution: $R_1 = \{c,f\}$, $R_2 = \{a,d\}$, and $R_3 = \{b,e,g\}$. Note that this register binding solution uses 3 registers (i.e., the minimum number of registers). Figure 4(a) displays the corresponding circuit graph G3. Then, for the circuit graph G3, we derive the corresponding constraint graph $G_{cg}(G3)$ shown in Figure 4(b). After the optimal clock skew scheduling is applied, we find that the smallest feasible clock period is 12, which is the lower bound of the clock period. The optimal clock skew schedule is $T_{host} = 0$, $T_1 = -4$, $T_2 = 4$, and $T_3 = -4$.

Based on the above observations, we find that different register binding solutions lead to different smallest feasible clock periods (achieved by the optimal clock skew scheduling). In this paper, we study the register binding for clock period minimization.

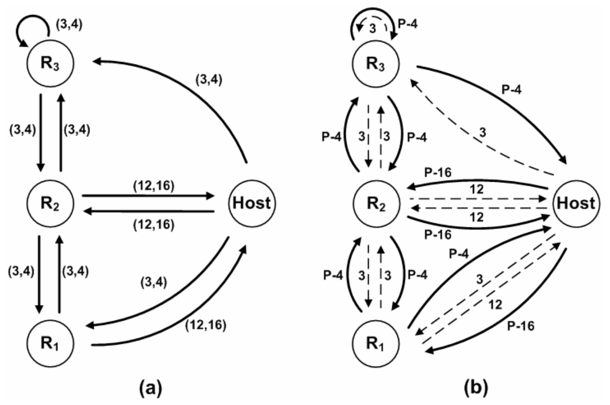
Figure 3: (a) Circuit graph G2. (b) Constraint graph $G_{cg}(G2)$.

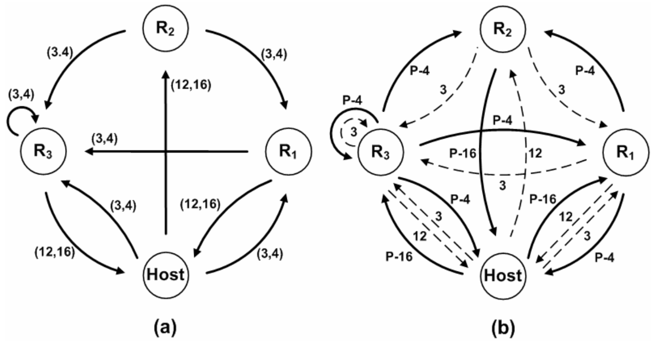
Figure 4: (a) Circuit graph G3. (b) Constraint graph $G_{cg}(G3)$.

5. MILP APPROACH

In this Section, we present an MILP approach to formally formulate our problem. Given a scheduled DFG and a constraint on the number of registers, our objective is to minimize the clock period via the application of simultaneous register binding and optimal clock skew scheduling. Note that our MILP formulation guarantees obtaining the optimal solution.

First, we introduce the variables used in our MILP formulation. For every combination of variable u (in the scheduled DFG) and register R_i, we define a binary variable $x_{u,i}$ (in our MILP formulation). If variable u is assigned to register R_i, the value of $x_{u,i}$ is 1; otherwise, the value of $x_{u,i}$ is 0. For each variable u (in the scheduled DFG), we define a real-value variable T_u (in our MILP formulation) that denotes the clock arrival time of variable u. For each register R_i, we define real-value variable T_i (in our MILP formulation) that denotes the clock arrival time of register R_i.

Suppose that we are given *n* registers, including R_1, R_2, …, and R_n. Then, we formally formulate our problem as below. The objective function is to minimize the clock period P. Note that each variable must be assigned to one register. Therefore, for each variable u, we have the following constraint:

$$\sum_{i=1}^{n} x_{u,i} = 1. \qquad \text{(Formula 1)}$$

If two variables have overlapping lifetimes, they cannot share the same register. Therefore, if both variable u and variable v have overlapping lifetimes, we have the following constraint for each register R_i: $x_{u,i} + x_{v,i} \le 1$. (Formula 2)

If variable u is assigned to register R_i (i.e., $x_{u,i} = 1$), then the value of T_u must be exactly the same as the value of T_i. On the other hand, if variable u is not assigned to register R_i (i.e., $x_{u,i} = 0$), then there is no constraint on the clock skew between T_u and T_i. Let *s* denote a constant value that approximates infinity, i.e., $s \rightarrow \infty$, and thus does not restrict the clock skew. Then, for every combination of variable u and register R_i, we have the following two constraints:

$T_u - T_i \le (1 - x_{u,i}) * s$. (Formula 3)

$T_i - T_u \le (1 - x_{u,i}) * s$. (Formula 4)

Let D_k denote the constant value that corresponds to the maximum delay of operation o_k. Then, for the input variable u and the output variable v of operation o_k, we have the following setup constraint:

$T_u - T_v \le P - D_k$. (Formula 5)

Let d_k denote the constant value that corresponds to the minimum delay of operation o_k. Then, for the input variable u and the output variable v of operation o_k, we have the following hold constraint:

$T_v - T_u \le d_k$. (Formula 6)

The number of constraints due to Formula 1, Formula 2, Formula 3, Formula 4, Formula 5, and Formula 6 are O(*m*), O(m^2), O(*mn*), O(*mn*), O(m^2), and O(m^2), respectively, where *m* is the number of variables in the scheduled DFG and *n* is the number of registers. Therefore, the number of constraints in our MILP formulation is O(m^2+*mn*).

Take the scheduled DFG shown in Figure 1(a) as an example. Suppose that we are given 3 registers, including R_1, R_2, and R_3. Our objective is to minimize the clock period P. Due to Formula 1, we have the following constraints:

$x_{a,1}+x_{a,2}+x_{a,3}=1$; $x_{b,1}+x_{b,2}+x_{b,3}=1$; $x_{c,1}+x_{c,2}+x_{c,3}=1$;
$x_{d,1}+x_{d,2}+x_{d,3}=1$; $x_{e,1}+x_{e,2}+x_{e,3}=1$; $x_{f,1}+x_{f,2}+x_{f,3}=1$;
$x_{g,1}+x_{g,2}+x_{g,3}=1$.

Due to Formula 2, we have the following constraints:

$x_{a,1}+x_{b,1}+x_{c,1}\le 1$; $x_{a,2}+x_{b,2}+x_{c,2}\le 1$; $x_{a,3}+x_{b,3}+x_{c,3}\le 1$;
$x_{c,1}+x_{d,1}+x_{e,1}\le 1$; $x_{c,2}+x_{d,2}+x_{e,2}\le 1$; $x_{c,3}+x_{d,3}+x_{e,3}\le 1$;
$x_{f,1}+x_{g,1}\le 1$; $x_{f,2}+x_{g,2}\le 1$; $x_{f,3}+x_{g,3}\le 1$.

Note that *s* is a constant value that does not restrict the clock skew. Since the clock skew is impossible greater than the multiplication of the longest combinational delay and the number of control steps, we let *s* be this value. Thus, we have *s* = 16 *4 = 64. Due to Formula 3 and Formula 4, we have the following constraints:

$T_a-T_1\le(1-x_{a,1})*64$; $T_1-T_a\le(1-x_{a,1})*64$; $T_a-T_2\le(1-x_{a,2})*64$;
$T_2-T_a\le(1-x_{a,2})*64$; $T_a-T_3\le(1-x_{a,3})*64$; $T_3-T_a\le(1-x_{a,3})*64$;
$T_b-T_1\le(1-x_{b,1})*64$; $T_1-T_b\le(1-x_{b,1})*64$; $T_b-T_2 \le(1-x_{b,2})*64$;
$T_2-T_b\le(1-x_{b,2})*64$; $T_b-T_3\le(1-x_{b,3})*64$; $T_3-T_b\le(1-x_{b,3})*64$;
$T_c-T_1\le(1-x_{c,1})*64$; $T_1-T_c\le(1-x_{c,1})*64$; $T_c-T_2\le(1-x_{c,2})*64$;
$T_2-T_c\le(1-x_{c,2})*64$; $T_c-T_3\le(1-x_{c,3})*64$; $T_3-T_c\le(1-x_{c,3})*64$;
$T_d-T_1\le(1-x_{d,1})*64$; $T_1-T_d\le(1-x_{d,1})*64$; $T_d-T_2\le(1-x_{d,2})*64$;
$T_2-T_d\le(1-x_{d,2})*64$; $T_d-T_3\le(1-x_{d,3})*64$; $T_3-T_d\le(1-x_{d,3})*64$;
$T_e-T_1\le(1-x_{e,1})*64$; $T_1-T_e\le(1-x_{e,1})*64$; $T_e-T_2\le(1-x_{e,2})*64$;
$T_2-T_e\le(1-x_{e,2})*64$; $T_e-T_3\le(1-x_{e,3})*64$; $T_3-T_e\le(1-x_{e,3})*64$;
$T_f-T_1\le(1-x_{f,1})*64$; $T_1-T_f\le(1-x_{f,1})*64$; $T_f-T_2\le (1-x_{f,2})*64$;
$T_2-T_f\le(1-x_{f,2})*64$; $T_f-T_3\le(1-x_{f,3})*64$; $T_3-T_f \le(1-x_{f,3})*64$;
$T_g-T_1\le(1-x_{g,1})*64$; $T_1-T_g\le(1-x_{g,1})*64$; $T_g-T_2\le(1-x_{g,2})*64$;
$T_2-T_g\le(1-x_{g,2})*64$; $T_g-T_3\le(1-x_{g,3})*64$; $T_3-T_g\le(1-x_{g,3})*64$.

Due to Formula 5, we have the following constraints:

$T_{host}-T_a\le P-16$; $T_{host}-T_b\le P-4$; $T_{host}-T_c\le P-4$; $T_{host}-T_d\le P-16$;
$T_a-T_e\le P-4$; $T_b-T_e\le P-4$; $T_c-T_g \le P-4$; $T_d-T_f\le P-4$;
$T_e-T_g\le P-4$; $T_f-T_{host}\le P-16$; $T_g-T_{host}\le P-16$.

Due to Formula 6, we have the following constraints:

$T_a-T_{host}\le 12$; $T_b-T_{host}\le 3$; $T_c-T_{host}\le 3$; $T_d-T_{host}\le 12$;
$T_e-T_a\le 3$; $T_e-T_b\le 3$; $T_g-T_c\le 3$; $T_f-T_d\le 3$;
$T_g-T_e\le 3$; $T_{host}-T_f\le 12$; $T_{host}-T_g\le 12$.

After solving the MILP formulation, we find that the minimum clock period P is 12. The binary variables $x_{c,1}$, $x_{f,1}$, $x_{a,2}$, $x_{d,2}$, $x_{b,3}$, $x_{e,3}$, $x_{g,3}$ are 1, and other binary variables are 0; thus, the register binding solution is R_1 = {c,f}, R_2 = {a,d}, and R_3 = {b,e,g}. The clock arrival times are as below: T_{host} = 0, $T_1 = T_c = T_f = -4$, $T_2 = T_a = T_d = 4$, and $T_3 = T_b = T_e = T_g = -4$.

6. HEURISTIC APPROACH

In this Section, we propose a heuristic algorithm. Our algorithm is a process of assigning variables to registers. To evaluate a partial binding solution, at the beginning of our algorithm, we have a constraint graph in which each vertex corresponds to an unassigned variable. In other words, an unassigned variable is treated as a virtual register. Furthermore, we use the notation dist(u,v) to denote the shortest distance from vertex u to vertex v. Given a constraint graph, if T_{host} is 0 and the clock period is the smallest feasible clock period, we can prove that: for each vertex u, the earliest possible clock arrival time E_u is -1*dist(u,host), and the latest possible clock arrival time L_u is dist(host,u). Then, for each vertex u in the constraint graph, we define that the width of possible clock arrival time W_u is L_u-E_u.

Figure 5 gives the pseudo code, where the notation *sche* denotes the given scheduled DFG, *n* denotes the given constraint on the number of registers, and the notation *sol* denotes our register binding solution. First, we derive the register binding solution that minimizes the number of registers as the reference solution. Assume that the number of registers in the reference solution is *p*. Clearly, the given constraint on the number of registers should be greater than or equal to *p*, i.e., $n \ge p$; otherwise, there is no feasible register binding solution. Then, we consider each register R_i, where i = 1, 2, …, and *p*. Among all the variables assigned to register R_i in the reference solution, we choose the variable that has smallest width of possible clock arrival time and assign it to register R_i in our solution *sol*. After that, we have assigned *p* variables to our solution *sol*.

In each loop of the while-do iteration, we choose an unassigned variable and assign it to a register. The iteration process repeats until all the variables are assigned to registers. The criteria in choosing an unassigned variable are as below.

(1) If an unassigned variable has only one register that can be assigned (due to the lifetime constraints), this variable has the highest priority.
(2) If both unassigned variables u and v have more than one register that can be assigned and W_u is smaller than W_v, then variable u has higher priority.

```
Procedure Heuristic(sche,n)
{ derive the reference solution (i.e., minimum-register solution);
  for each register R_1 (from register R_1 to register R_p) do
    { choose variable u that has smallest width of possible clock
        arrival time from register R_i in the reference solution;
        assign variable u to register R_i in our solution sol; }
  while (there still exists any unassigned variable) do
    { choose the highest priority unassigned variable u;
      assign variable u to the highest priority register R_i
      (among the n registers) in our solution sol; }
  return(sol); }
```

Figure 5: Pseudo code of our heuristic algorithm.

The total possible clock arrival time width TW of a constraint graph is defined as the summation of W_u of each vertex u in the constraint graph. Consider that an unassigned variable is chosen and it can be assigned to both register R_i and register R_j. Suppose that, if the variable is assigned to register R_i, the smallest feasible clock period becomes P_i and the total possible clock arrival time width becomes TW_i; if the variable is assigned to register R_j, the smallest feasible clock period becomes P_j and the total possible clock arrival time width becomes TW_j. We say that register R_i has higher priority, if one of the following two conditions are met:

(1) $P_i < P_j$.

(2) $P_i = P_j$ and $TW_i < TW_j$.

The number of while-do iterations is at most *m*, where *m* is the number of variables in the scheduled DFG *sche*. The time complexity spent in assigning the highest priority variable to the highest priority register is $O(n^*|T_{OCSS}|)$, where $|T_{OCSS}|$ is the time complexity of optimal clock skew scheduling used. Therefore, the time complexity of our heuristic algorithm is $O(m^*n^*|T_{OCSS}|)$.

Take the scheduled DFG shown in Figure 1(a) as an example. Suppose that the given constraint on the number of registers is 3. Initially, all the variables are unassigned. Based on the constraint graph shown in Figure 6(a), we find that the smallest feasible clock period is 12. By assuming that $T_{host} = 0$ and the clock period P is 12, we have $E_a = 4$, $L_a = 12$, $E_b = -7$, $L_b = 3$, $E_c = -8$, $L_c = 3$, $E_d = 4$, $L_d = 4$, $E_e = -4$, $L_e = 4$, $E_f = -4$, $L_f = -4$, $E_g = -12$, and $L_g = -4$. As a result, we have $W_a = 8$, $W_b = 10$, $W_c = 11$, $W_d = 0$, $W_e = 8$, $W_f = 0$, and $W_g = 8$.

We adopt the register binding solution obtained by the left edge algorithm as the reference solution. Since variable c and variable f are assigned to register R_1 in the reference solution and W_f is smaller than W_c, variable f is assigned to register R_1 in our solution *sol*. Since variable a, variable d, and variable g are assigned to register R_2 in the reference solution and W_d is smaller than both W_a and W_g, variable d is assigned to register R_2 in our solution *sol*. Since variable b and variable e are assigned to register R_3 in the reference solution and W_e is smaller than W_b, variable e is assigned to register R_3 in our solution *sol*. Thus, our solution *sol* becomes $R_1 = \{f\}$, $R_2 = \{d\}$, and $R_3 = \{e\}$.

The remaining unassigned variables are a, b, c, and g. Due to the lifetime constraints, variable c only can be assigned to register R_1. Thus, variable c is chosen and assigned to register R_1. Thus, our solution *sol* becomes $R_1 = \{c,f\}$, $R_2 = \{d\}$, and $R_3 = \{e\}$. Figure 6(b) gives the constraint graph. The smallest feasible clock period is still 12. In this constraint graph, we have $W_a = 8$, $W_b = 10$, $W_g = 8$, $W_1 = 0$, $W_2 = 0$, and $W_3 = 8$.

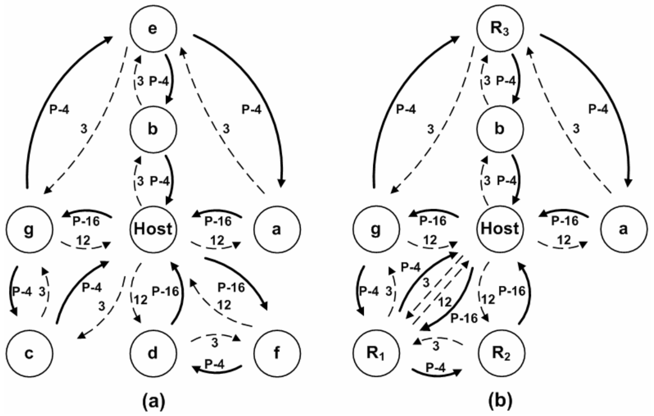

Figure 6: Snapshots of our heuristic algorithm.

Variables a, b, and g have more than one register that can be assigned. Note that W_a and W_g are equal, while W_b is larger. Thus, we can choose variable a or variable g. Suppose that variable a is chosen (in fact, in this example, no matter variable a or variable g is chosen, the same final register binding solution is obtained). We find that variable a can be assigned to register R_2 and register R_3. If variable a is assigned to register R_2, the smallest feasible clock period becomes 12 and the total possible clock arrival time width becomes 26 (since W_b, W_g, W_1, W_2, and W_3 become 10, 8, 0, 0, and 8, respectively); if variable a is assigned to register R_3, the smallest feasible clock period becomes 12 and the total possible clock arrival time width becomes 2 (since W_b, W_g, W_1, W_2, and W_3 become 2, 0, 0, 0, and 0, respectively). Thus, variable a is assigned to register R_2.

Due to the lifetime constraints, variable b only can be assigned to register R_3. Thus, variable b is chosen and assigned to register R_3. Finally, we consider variable g. Variable g can be assigned to register R_2 and register R_3. If variable g is assigned to register R_2, the smallest feasible clock period becomes 16; if variable g is assigned to register R_3, the smallest feasible clock period becomes 12. Thus, variable g is assigned to register R_3. As a result, our solution *sol* becomes $R_1 = \{c,f\}$, $R_2 = \{a,d\}$, and $R_3 = \{b,e,g\}$. Note that, in this example, our heuristic approach achieves the same solution as our MILP approach.

7. EXPERIMENTAL RESULTS

We implement our approach on a personal computer with AMD K8-3GHz CPU and 512M Bytes RAM. We use Extended LINGO Release 8.0 as the MILP solver. As an alternative, we also use C++ programming language to implement our heuristic algorithm.

Seven benchmark circuits are used to test the effectiveness of our approach. Benchmark circuits HAL, AR, Bandpass Filter (BF), and Elliptic Wave Filter (EWF) are popular DSP applications and widely used in the high-level synthesis community [5,8], while benchmark circuits IDCT1, Motion, and Sha1 are representative functions adopted from the MediaBench suite [9]. The scheduled DFGs of these benchmark circuits are derived by the scheduling approach proposed in [8].

Without loss of generality, we assume that all the functional units and all the variables (registers) are 8-bit designs. The modules in the Synopsys DesignWare are adopted to implement the following functional units: ALU, multiplier, divisor, multiplexer, and comparator. Moreover, these functional units are targeted to Artisan UMC 0.18μm standard cell library. The maximum delay (minimum delay) of ALU, multiplier, divisor, multiplexer, and comparator are 2.20 ns (1.82 ns), 4.70 ns (0.67 ns), 11.90 ns (0.85 ns), 0.68 ns (0.54 ns), and 1.41 ns (1.13 ns), respectively.

Table 1 tabulates the characteristics of benchmark circuits. The column *#var* gives the number of variables. The column *Design Constraints* gives 6-tuple (*#steps*,*#alu*,*#mul*,*#div*,*#mux*,*#comp*), where *#steps*, *#alu*, *#mul*, *#div*, *#mux*, and *#comp* are the number of control steps, ALUs, multipliers, divisors, multiplexers, and comparators, respectively. For example, the design constraints of benchmark circuit HAL include 4 control steps, 2 ALUs, 2 multipliers, and 1 comparator. The column *Left Edge* describes the register binding solution obtained by the left edge algorithm. The column R_B denotes the minimum number of registers for a feasible register binging solution. The column P_{OCSS} denotes the smallest feasible clock period obtained by the optimal clock skew scheduling. The column P_B gives the lower bound of the clock period (i.e., each register represents only one variable).

Table 2 tabulates our experimental results, including our MILP approach and our heuristic approach. The column *#reg* describes the given constraint on the number of registers. The column P_{OCSS} denotes the smallest feasible clock period obtained by the optimal clock skew scheduling. Note that our MILP approach guarantees achieving the optimal solution under the given constraint on the number of registers. The column *CPU Time* denotes the CPU time in seconds. From Table 2, we have the following observations:

(1) In the benchmark circuits HAL, AR, BF, EWF, Motion, and Sha1, our MILP approach finds register binding solutions that not only use the minimum number of registers but also work with the lower bound of the clock period.
(2) In the benchmark circuit IDCT1, our MILP approach shows that: the lower bound of the clock period can be achieved with a very small overhead on the number of registers. Compared with the minimum number of registers for a feasible register binding solution, only an extra register is needed for the lower bound of the clock period.
(3) Compared with our MILP approach, we find that our heuristic approach achieves the near-optimal solutions. Therefore, our heuristic approach provides a good alternative to solve the problem in polynomial time complexity.

8. CONCLUSIONS

This paper is the first attempt to the high-level synthesis of non-zero clock skew circuits. We show that the register binding has a significant impact on the utilization of clock skew. Based on that observation, we propose an approach to obtain the minimum-period register binding solution. In most benchmark circuits, the lower bound of the clock period can be achieved without any extra overhead on the number of registers.

In fact, the clock skew can also be utilized to improve the circuit reliability [3]. This extension of our approach is straightforward.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Council of R.O.C. under grant number NSC 94-2220-E-033-001.

REFERENCES

[1] J.P. Fishburn, “Clock Skew Optimization,” IEEE Trans. on Computers, vol. 39, no. 7, pp. 945—951, 1990.

[2] R.B. Deokar and S.S. Sapatnekar, “A Graph-Theoretic Approach to Clock Skew Optimization,” Proc. of IEEE International Symposium on Circuits and Systems, vol. 1, pp. 407—410, 1994.

[3] C. Albrecht, B. Korte, J. Schietke, and J. Vygen, “Cycle Time and Slack Optimization for VLSI Chips,” Proc. of IEEE/ACM International Conference on Computer Aided Design, pp. 232—238, 1999.

[4] F.J. Kurdahi and A.C. Parker, “REAL: A Program for Register Allocation,” Proc. of IEEE/ACM Design Automation Conference, pp. 210—215, 1987.

[5] C. Tseng and D.P. Siewiorek, “Automatic Synthesis of Data Paths in Digital Systems,” IEEE Trans. on Computer-Aided Design,” vol. 5, no. 3, pp. 379—395, 1986.

[6] L. Zhong, J. Luo, Y. Fei, and N.K. Jha, “Register Binding Based Power Management for High-Level Synthesis of Control-Flow Intensive Behaviors,” Proc. of IEEE International Conference on Computer Design, pp. 391—394, 2002.

[7] D. Chen and J. Cong, “Register Binding and Port Assignment for Multiplexer Optimization,” Proc. of IEEE/ACM Asia and South Pacific Design Automation Conference, pp. 68—73, 2004.

[8] S.H. Huang and C.H. Cheng, “A Formal Approach to the Slack Driven Scheduling Problem in High Level Synthesis,” Proc. of IEEE International Symposium on Circuits and Systems, pp. 5633—5636, 2005.

[9] C. Lee, M. Potkonjak, and W.H. Maggione-Smith, “MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems,” Proc. of IEEE International Symposium on Microarchitecture, pp. 330—335, 1997.

Circuit	#var	Design Constraints	Left Edge		P_B (ns)
			R_B	P_{OCSS} (ns)	
HAL	8	(4,2,2,0,0,1)	4	4.70	4.03
AR	30	(8,4,3,0,0,0)	8	4.70	4.03
BF	30	(8,3,2,0,0,0)	6	4.48	4.03
EWF	47	(14,4,2,0,0,0)	11	4.70	4.03
IDCT1	60	(12,6,3,1,0,0)	21	11.90	11.05
Motion	430	(42,12,15,8,2,0)	111	11.90	11.05
Sha1	540	(86,41,34,0,0,0)	100	4.70	4.03

Table 1: Characteristics of benchmark circuits.

Circuit	#reg	Our MILP		Our Heuristic	
		P_{OCSS} (ns)	CPU Time (s)	P_{OCSS} (ns)	CPU Time (s)
HAL	4	4.03	< 1	4.03	< 1
AR	8	4.03	2	4.03	1
BF	6	4.03	2	4.37	1
	7	4.03	3	4.03	2
EWF	11	4.03	4	4.30	3
	12	4.03	5	4.03	4
IDCT1	21	11.90	17	11.90	10
	22	11.05	17	11.05	10
Motion	111	11.05	1782	11.90	865
	112	11.05	1846	11.05	884
Sha1	100	4.03	1848	4.37	955
	102	4.03	1889	4.03	961

Table 2: Our experimental results.