

Power-Management Scheduling for Peak Power Minimization*

SHIH-HSU HUANG AND CHUN-HUA CHENG

*Department of Electronic Engineering
Chung Yuan Christian University
Chungli, 320 Taiwan*

As the design complexity continues to increase, huge peak power has become an important concern. A widely used power management technique is to shut down unused operations. However, an unused operation cannot be shut down, unless all the operations involved in identifying the control/data flow of this operation have been scheduled before this operation. Therefore, operation scheduling has a significant impact on the potential of power management. Based on that observation, in this paper, we study the simultaneous application of operation scheduling and power management for peak power minimization. Our work includes the following two aspects. First, we propose an integer linear program to formally formulate the problem. Second, we propose a heuristic algorithm to solve the problem in polynomial time complexity. Compared with previous work, benchmark data show that our approach can significantly reduce the peak power.

Keywords: electronic design automation, high-level synthesis, operation scheduling, power management, peak power, integer linear programming

1. INTRODUCTION

In high-level synthesis [1], a behavior-level circuit design is represented by a control/data flow graph (CDFG). The task of *operation scheduling* [2-14] is to assign each operation in the CDFG to a specific control step to start its execution. It has been recognized that operation scheduling greatly influences all quality aspects of the final implementation. Therefore, according to Gajski [1], operation scheduling is “perhaps the most important task in high-level synthesis”.

The peak power is the maximum power consumption of the circuit design at any instant during its execution. Huge peak power may cause many reliability problems, such as large voltage drop, large current density, and large heat dissipation. As the design complexity continues to increase, there is a demand to reduce the peak power. However, most previous operation scheduling algorithms [2-14] focus on the minimization of delay, area, or average power consumption. Up to now, Shiue’s work [15] is the only paper that studies the problem of operation scheduling for peak power minimization. Under the design constraints (*i.e.*, timing and resource constraints), Shiue [15] formally formulates the peak power minimization problem as an integer linear program.

Shiue’s work [15] assumes that all the operations in the CDFG must be executed. However, a CDFG may include many conditionals. For each conditional, there are some

Received March 2, 2007; revised July 4 & October 19, 2007; accepted October 25, 2007.

Communicated by Chung-Ping Chung.

* A preliminary version, entitled “Peak power minimization through power management scheduling,” has appeared in Proceedings of IEEE International Symposium on Asia Pacific Conference on Circuits and Systems (APCCAS), 2006. This work was supported in part by the National Science Council of Taiwan, R.O.C., under grant No. NSC 93-2220-E-033-001.

operations whose outputs are not used; in other words, for each conditional, there are some operations that are not necessary to be executed. In recent years, shutting down unused operations [7, 10, 13] has become a widely used power management technique. Since Shiu's work [15] does not shut down unused operations, the peak power can be further reduced.

Intuitively, we can use a two-step process for peak power minimization: operation scheduling followed by power management. However, we cannot shut down an unused operation, unless all the operations involved in identifying the control/data flow of this operation have been scheduled at least one control step before this operation. As a result, the two-step process cannot effectively reduce the peak power. Since operation scheduling has a significant impact on the potential of power management, the consideration of power management should be integrated into the task of operation scheduling. Based on that observation, in this paper, we study the simultaneous application of operation scheduling and power management for peak power minimization.

To the best of our knowledge, this paper is the first work that deals with the problem of simultaneous operation scheduling and power management for peak power minimization. Our work includes the following two aspects. First, we use an integer linear program to formally formulate this problem. Note that our integer linear program guarantees solving this problem optimally. Second, we propose a heuristic algorithm to solve this problem in polynomial time complexity. Benchmark data show that our heuristic algorithm achieves near-optimal solutions within a small CPU time.

Compared with previous works [7, 10, 13, 15], the main distinctions of our work as below.

- (1) In [7, 10, 13], they shut down unused operations for reducing the average power consumption. They [7, 10, 13] do not deal with the peak power minimization problem. On the other hand, in this paper, we shut down unused operations for reducing the peak power consumption. This paper does not deal with the average power minimization problem.
- (2) Different from Shiu's work [15], our integer linear program integrates power management into operation scheduling. It is noteworthy to mention that, due to power management, the calculation of power consumption at each control step becomes more complex than Shiu's work [15]. Therefore, our extension is not straightforward.

The organization of this paper is as below. Section 2 provides the background materials, and section 3 gives a motivational example. Section 4 proposes our ILP (integer linear programming) approach. Section 5 presents our heuristic algorithm. Section 6 demonstrates our experimental results. Finally, some concluding remarks are given in section 7.

2. PRELIMINARIES

In section 2.1, we provide the background materials for operation scheduling [2-15]. In section 2.2, we provide the background materials for power management [7, 10, 13].

2.1 Operation Scheduling

In high-level synthesis, a behavior-level circuit design is represented by a CDFG, where each node corresponds to an operation, and each directed edge corresponds to a dependency relationship (data dependency or control dependency). In the CDFG, each conditional is composed of a comparison operation, a multiplexing operation, and a directed edge (i.e., a control dependency) from the comparison operation to the multiplexing operation.

We use the behavior description shown in Fig. 1 (a) for illustration. The behavior description can be represented by the CDFG depicted in Fig. 1 (b). In this CDFG, there are nine operations: $o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8,$ and o_9 . The statement $m8 = x3 * x4$ corresponds to operation o_8 , the statement $m9 = x5 + x6$ corresponds to operation o_9 , the statement $m6 = m8 * x7$ corresponds to operation o_6 , the statement $m7 = m8 + x8$ corresponds to operation o_7 , and the statement $m3 = m8 + x9$ corresponds to operation o_3 . There are two conditional statements: the conditional statement if ($x1 > x2$) is represented by comparison operation o_5 , multiplexing operation o_4 , and directed edge from operation o_5 to operation o_4 , and the conditional statement if ($m9 > x10$) is represented by comparison operation o_2 , multiplexing operation o_1 , and directed edge from operation o_2 to operation o_1 .

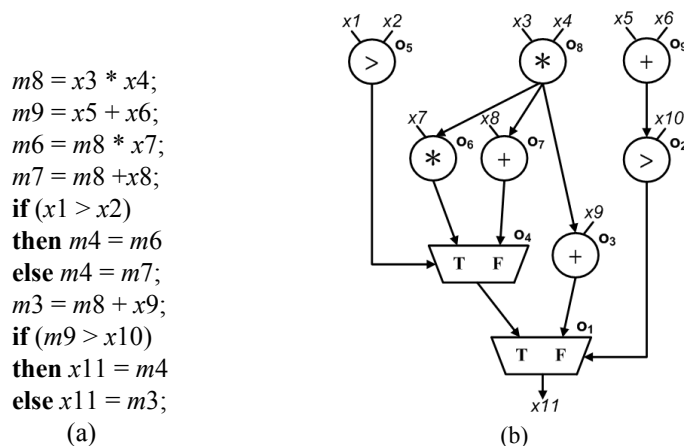


Fig. 1. (a) A behavior description; (b) A CDFG.

Given the design constraints (timing and resource constraints), the task of operation scheduling is to assign each operation in the CDFG to a specific control step to start its execution. We say that a CDFG is a scheduled CDFG if operation scheduling has been performed. Once the operations are scheduled, the number and types of functional units and the lifetimes of variables are fixed. Therefore, operation scheduling is “perhaps the most important task in high-level synthesis” [1].

The list scheduling technique [2] has been adopted in many high-level synthesis systems. The operations in the CDFG are assigned to control steps from the first control step to the last. An operation is called ready operations if all of its predecessors have

been executed. Note that ready operations can be scheduled into the current control step. However, if the number of scheduled ready operations has reached the number of resources, the remaining ready operations should be postponed to the next control step.

Take the CDFG shown in Fig. 1 (b) for illustration. Assume that the delay of each operation is one control step. Under the constraints that four control steps, one multiplier (for the execution of multiplication operations), two adders (for the execution of addition operations), one comparator (for the execution of comparison operations), and one multiplexer (for the execution of multiplexing operations), we apply list scheduling [2] to it. As a result, we have the scheduled CDFG as shown in Fig. 2. Since operation scheduling does not need the information of variable names, for brevity sake, we do not provide variable names in Fig. 2.

2.2 Power Management

Suppose that the power consumptions of multiplier, adder, comparator, and multiplexer, are 20, 4, 3, and 1, respectively. We analyze the peak power of the scheduled CDFG shown in Fig. 2 as below. The power consumptions at control step 1, control step 2, control step 3, and control step 4 are 27 (*i.e.*, $3 + 20 + 4 = 27$), 31 (*i.e.*, $20 + 4 + 4 + 3 = 31$), 1, and 1, respectively. Therefore, the peak power is 31.

However, in fact, due to the conditionals, the outputs of some operations are not used under certain situations. Therefore, if unused operations can be shut down by adding extra control logics, the peak power may be reduced.¹ Take the scheduled CDFG shown in Fig. 2 for example. For example, operation o_6 and operation o_7 are mutually exclusive. If the result of comparison operation o_5 is 0 (F), the output of multiplication operation o_6 is not used; thus, we can shut down multiplication operation o_6 when the result of comparison operation o_5 is 0 (F). On the other hand, if the result of comparison operation o_5 is 1 (T), the output of addition operation o_7 is not used; thus, we can shut down addition operation o_7 when the result of comparison operation o_5 is 1 (T).

Note that, the result of a comparison operation cannot be used to shut down an unused operation, unless the comparison operation completes its execution before the execution of the unused operation. Let's use the scheduled CDFG shown in Fig. 2 as an example. Since comparison operation o_5 completes its execution before the execution of addition operation o_7 , we can use the result of comparison operation o_5 to shut down addition operation o_7 . On the other hand, since comparison operation o_2 and addition operation o_7 are executed simultaneously, we cannot use the result of comparison operation o_2 to shut down operation o_7 .

Intuitively, for reducing the peak power, we should shut down as many unused operations as possible. Using the scheduled CDFG shown in Fig. 2 as an example, we have the following two power managements.

- (1) Multiplication operation o_6 can be shut down when the result of comparison operation o_5 is 0.
- (2) Addition operation o_7 can be shut down when the result of comparison operation o_5 is 1.

¹ Since the power consumptions of extra control logics (for shutting down unused operations) are small, they are ignored in previous works [7, 10, 13]. Following the same assumption in previous works [7, 10, 13], here, we also ignore the power consumptions of extra control logics (for shutting down unused operations). Note that the implementation of control logics is discussed in the Appendix.

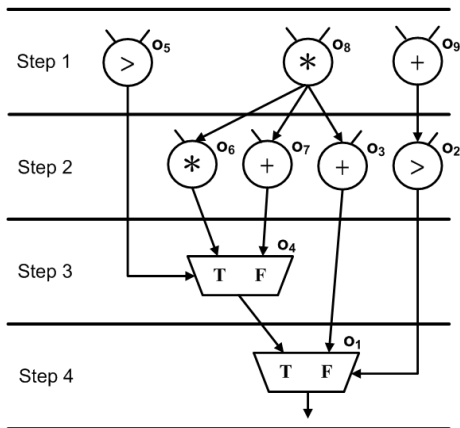


Fig. 2. Scheduled CDFG obtained by list scheduling.

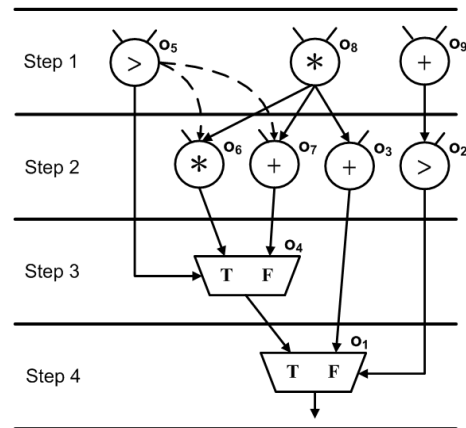


Fig. 3. A modified scheduled CDFG (*i.e.*, a scheduled CDFG with power management).

Fig. 3 gives the modified scheduled CDFG (*i.e.*, the scheduled CDFG with power management). There are two extra directed edges (in dotted lines) used to represent the two power managements: an extra directed edge from comparison operation o_5 to multiplication operation o_6 , and an extra directed edge from comparison operation o_5 to addition operation o_7 . It is noteworthy to mention that these extra directed edges correspond to extra control dependencies. In [7, 10, 13], these extra directed edges are referred to as soft edges.

Due to power management, the calculation of power consumption at each control step becomes more complex. For the convenience of readers, we analyze the scheduled CDFG shown in Fig. 3 as below.

- (1) Control step 1. Since comparison operation o_5 , multiplier operation o_8 , and addition operation o_9 are executed, the power consumption is 27 (*i.e.*, $3 + 20 + 4 = 27$).
- (2) Control step 2. If the result of comparison operation o_5 is 0, since addition operation o_7 , addition operation o_3 , and comparison operation o_2 are executed, the power consumption is 10 (*i.e.*, $4 + 4 + 3 = 11$). If the result of comparison operation o_5 is 1, since multiplication operation o_6 , addition operation o_3 , and comparison operation o_2 are executed, the power consumption is 27 (*i.e.*, $20 + 4 + 3 = 27$).
- (3) Control step 3. Since multiplexing operation o_4 is executed, the power consumption is 1.
- (4) Control step 4. Since multiplexing operation o_1 is executed, the power consumption is 1.

Based on the above analysis, we find that the peak power of the scheduled CDFG shown in Fig. 3 is 27. Compared with the original scheduled CDFG shown in Fig. 2, the peak power is reduced from 31 to 27.

3. MOTIVATION

Intuitively, we can use a two-step process for peak power minimization: operation scheduling followed by power management. However, the two-step process cannot mini-

mize the peak power. Take the CDFG shown in Fig. 1 (b) for illustration. Suppose that the design constraints (*i.e.*, timing and resource constraints) are four control steps, one multiplier, two adders, one comparator, and one multiplexer. For generality, in the following, we use list scheduling [2] and Shiue's work [15] as the operation scheduling algorithm, respectively.

- (1) Suppose that we use list scheduling as the operation scheduling algorithm. After list scheduling is applied, we obtain the scheduled CDFG as shown in Fig. 2. Then, based on the scheduled CDFG shown in Fig. 2, we attempt to shut down as many unused operations as possible. We find that two soft edges can be added. As a result, we have the scheduled CDFG as shown in Fig. 3. Therefore, if we use list scheduling as the operation scheduling algorithm, the peak power of the two-step process is 27.
- (2) Suppose that we use Shiue's work as the operation scheduling algorithm. After Shiue's work is applied, we obtain the scheduled CDFG as shown in Fig. 4. Then, based on the scheduled CDFG shown in Fig. 4, we attempt to shut down as many unused operations as possible. We find that no soft edges can be added. Therefore, if we use Shiue's work as the operation scheduling algorithm, the peak power of the two-step process is also 27.

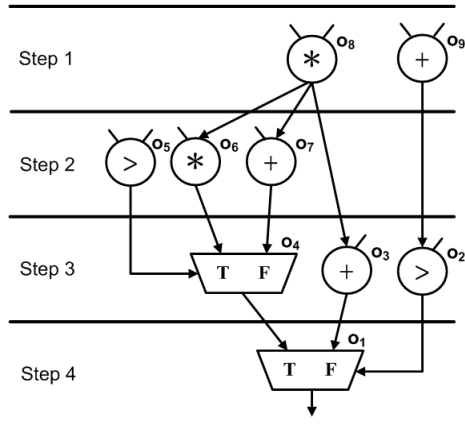


Fig. 4. Scheduled CDFG obtained by Shiue's work.

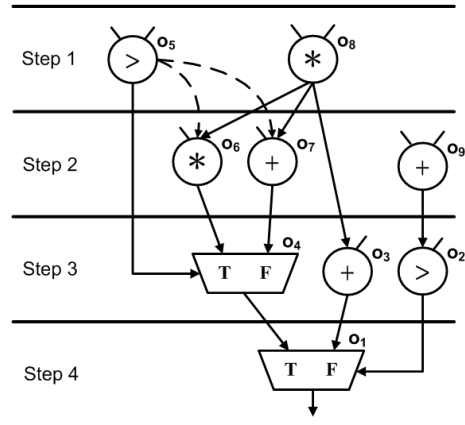


Fig. 5. Our scheduled CDFG.

From the above discussions, we know that: no matter we use list scheduling or Shiue's work as the operation scheduling algorithm, the peak power of the two-step process is 27. However, in fact, there exists a scheduled CDFG whose peak power is less than 27. Consider the scheduled CDFG shown in Fig. 5. There are two soft edges: a soft edge from comparison operation o_5 to multiplication operation o_6 , and a soft edge from comparison operation o_5 to addition operation o_7 . We analyze the power consumption of each control step as below.

- (1) Control step 1. Since comparison operation o_5 and multiplier operation o_8 are executed, the power consumption is 23 (*i.e.*, $3 + 20 = 23$).
- (2) Control step 2. If the result of comparison operation o_5 is 0, since addition operation

- o_7 and addition operation o_9 are executed, the power consumption is 8 (*i.e.*, $4 + 4 = 8$). If the result of comparison operation o_5 is 1, since multiplication operation o_6 and addition operation o_9 are executed, the power consumption is 24 (*i.e.*, $20 + 4 = 24$).
- (3) Control step 3. Since multiplexing operation o_4 , addition operation o_3 , and comparison operation o_2 are executed, the power consumption is 8 (*i.e.*, $.1 + 4 + 3 = 8$).
- (4) Control step 4. Since multiplexing operation o_1 is executed, the power consumption is 1.

Based on the above analysis, we find that: the peak power of the scheduled CDFG shown in Fig. 5 is only 24. Therefore, the two-step process does not minimize the peak power.

We point out the drawbacks of the two-step process as below.

- (1) At the stage of operation scheduling, the effect of power management is not taken into account. Due to the lack of the knowledge of power management, the peak power cannot be accurately calculated at the stage of operation scheduling.
- (2) An unused operation cannot be shut down, unless all the operations involved in identifying the control/data flow of this operation have been scheduled at least one control step before this operation. Since power management is not considered at the stage of operation scheduling, the result of operation scheduling often limits the application of power management.

To overcome these drawbacks, the power management should be taken into account at the stage of operation scheduling. Therefore, in this paper, we study the simultaneous application of operation scheduling and power management for peak power minimization. To the best of our knowledge, our paper is the first work that deals with this problem.

4. ILP APPROACH

In this section, we use ILP to formally formulate the simultaneous application operation scheduling and power management. Our objective is to minimize the peak power under the design constraints. Note that our ILP approach guarantees minimizing the peak power.

First, we introduce the notations used in our ILP approach as below.

- (1) The notation t denotes the number of control steps.
- (2) The notation $x_{i,j}$ denotes a binary variable (*i.e.*, an 0-1 integer variable). Binary variable $x_{i,j} = 1$, if and only if operation o_i is scheduled into control step j ; otherwise, binary variable $x_{i,j} = 0$.
- (3) The notation C_i denotes the set of comparison operations that can shut down operation o_i when operation o_i is unused. Note that we can determine the set C_i of each operation o_i from the CDFG.
- (4) The value $|A|$ denotes the number of elements in the set A .
- (5) The notation w_i is a constant value, which corresponds to the power consumption of operation o_i . Note that, in previous works [7, 10, 13, 15], they also assume that the power consumption of each operation is a constant value.

- (6) The notation $Y_{c,i}$ denotes a binary variable (*i.e.*, an 0-1 integer variable). Binary variable $Y_{c,i} = 1$, if and only if there is a soft edge from comparison operation o_c to operation o_i ; otherwise, binary variable $Y_{c,i} = 0$.
- (7) The notation $Z_{c,i,j}$ denotes a binary variable (*i.e.*, an 0-1 integer variable). Binary variable $Z_{c,i,j} = 1$, if and only if there is a soft edge from comparison operation o_c to operation o_i and operation o_i is scheduled into control step j ; otherwise, binary variable $Z_{c,i,j} = 0$.
- (8) The notation $Y_{A,i}$ denotes a binary variable (*i.e.*, an 0-1 integer variable). Binary variable $Y_{A,i} = 1$, if and only if, for each comparison operation in the set A , there is a soft edge from it to operation o_i ; otherwise, binary variable $Y_{A,i} = 0$.
- (9) The notation $Z_{A,i,j}$ denotes a binary variable (*i.e.*, an 0-1 integer variable). Binary variable $Z_{A,i,j} = 1$, if and only if operation o_i is scheduled into control step j , and for each comparison operation in the set A , there is a soft edge from it to operation o_i ; otherwise, binary variable $Z_{A,i,j} = 0$.
- (10) The notation H_s denotes the set of unused operations under a specific situation s .
- (11) The notation D_i is a constant, which corresponds to the delay of operation o_i (in terms of the number of control steps).
- (12) The notation E_i is a constant, which corresponds to the earliest possible control step of operation o_i . Note that we can use the ASAP calculation [4] to determine the value E_i of each operation o_i .
- (13) The notation L_i is a constant, which corresponds to the latest possible control step of operation o_i . Note that we can use the ALAP calculation [4] to determine the value L_i of each operation o_i .
- (14) The notation FU_k denotes the function unit k . We say that $o_i \in FU_k$, if and only if operation o_i can be executed by the function unit FU_k .
- (15) The notation M_k is a constant, which corresponds to the number of function unit k .
- (16) The notation $peak_power$ denotes a variable, which corresponds to the amount of peak power.

We formally formulate our problem as below. The objective function is to minimize the value of variable $peak_power$. Note that every operation must be scheduled into a control step. Therefore, for each operation o_i , we have the following constraint:

$$\sum_{j=E_i}^{L_i} x_{i,j} = 1. \quad (\text{Formula 1})$$

Each dependency relationship in the CDFG must be preserved. Therefore, for each dependency relationship $o_i \rightarrow o_p$ in the CDFG, we have the following formulation:

$$\sum_{j=E_i}^{L_i} (j + D_i - 1) \cdot x_{i,j} < \sum_{j=E_p}^{L_p} j \cdot x_{p,j}. \quad (\text{Formula 2})$$

The number of function unit k is M_k . Therefore, for each control step j and each function unit k , we have the following constraint:

$$\sum_{o_i \in FU_k} \sum_{E_i + D_i - 1 \geq j} x_{i,j} \leq M_k. \tag{Formula 3}$$

If the result of comparison operation o_c is used to shut down operation o_i , a soft edge is added into the CDFG. Note that a soft edge corresponds to a control dependency. Therefore, for each pair of comparison operation o_c and operation o_i , we have the following constraint:

$$\sum_{j=E_c}^{L_c} (j + D_c - 1) \cdot x_{c,j} < \sum_{j=E_i}^{L_i} j \cdot x_{p,j} + (1 - Y_{c,i}) \cdot t. \tag{Formula 4}$$

Suppose that the set $A \subseteq C_i$. Binary variable $Y_{A,i} = 1$, if for each comparison operation $o_c \in A \subseteq C_i$, binary variable $Y_{c,i} = 1$. Therefore, we have the following constraint:

$$\sum_{o_c \in A} Y_{c,i} \leq Y_{A,i} + |A| - 1. \tag{Formula 5}$$

Suppose that the set $A \subseteq C_i$. Binary variable $Y_{A,i} = 0$, if there exists a comparison operation $o_c \in A \subseteq C_i$ and binary variable $Y_{c,i} = 0$. Therefore, we have the following constraint:

$$Y_{A,i} \leq Y_{c,i}. \tag{Formula 6}$$

If binary variable $Y_{A,i} = 1$ and binary variable $x_{i,j} = 1$, then binary variable $Z_{A,i,j} = 1$. Therefore, we have the following constraint:

$$Y_{A,i} + x_{i,j} \leq Z_{A,i,j} + 1. \tag{Formula 7}$$

If binary variable $Y_{A,i} = 0$, then binary variable $Z_{A,i,j} = 0$. Therefore, we have the following constraint:

$$Z_{A,i,j} \leq Y_{A,i}. \tag{Formula 8}$$

If binary variable $x_{i,j} = 0$, then binary variable $Z_{A,i,j} = 0$. Therefore, we have the following constraint:

$$Z_{A,i,j} \leq x_{i,j}. \tag{Formula 9}$$

At each control step j , different situations lead to different unused operations. For a specific situation s , an unused operation $o_i \in H_s$ can be shut down, if there exist a comparison operation $o_c \in C_i$ and binary variable $Z_{c,i,j} = 1$. However, it is noteworthy to mention that, if more than one comparison operations in the set C_i can shut down operation o_i , the power saving should only be counted one time. Therefore, the power saving caused by shutting down an unused operation $o_i \in H_s$ is $\sum_{o_i \in H} \sum_{A \subseteq C_i} (-1)^{|A|} \cdot w_i \cdot Z_{A,i,j}$. As a result, we have the following constraint:

$$\sum_{i=1}^n x_{i,j} \cdot w_i + \sum_{i=1}^n \sum_{o_i \in H_s} \sum_{A \subseteq C_i} (-1)^{|A|} \cdot w_i \cdot Z_{A,i,j} \leq \text{peak_power}. \quad (\text{Formula } 10)$$

In the following, we use the CDFG shown in Fig. 1 (b) to illustrate our ILP approach. Suppose that the timing constraints are four control steps. Note that, according to the ASAP and ALAP schedules [4], we can prune redundant binary variables. Figs. 6 (a) and (b) give the ASAP and ALAP schedules of this CDFG. Table 1 gives all the binary variables (irredundant binary variables) associated with each operation. Table 2 gives all the binary variables associated with each set of comparison operations.

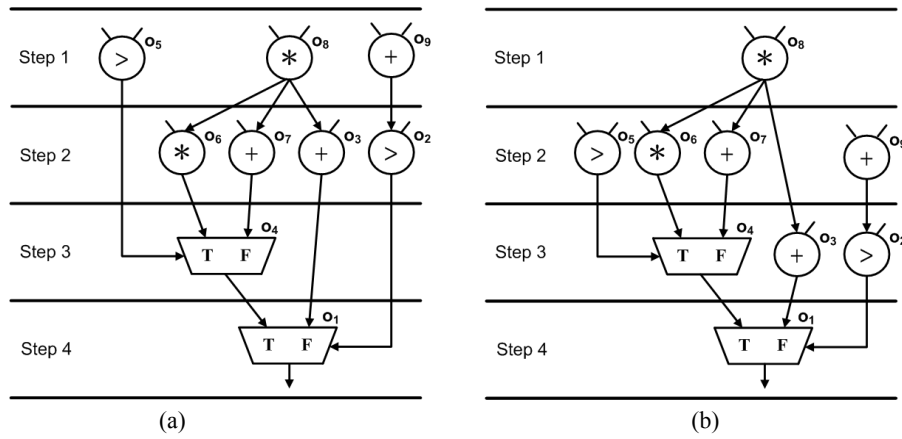


Fig. 6. (a) ASAP schedule; (b) ALAP schedule.

Table 1. Binary variables associated with each operation.

Operation	Associated Binary Variables
o_1	$x_{1,4}$
o_2	$x_{2,2}, x_{2,3}$
o_3	$x_{3,2}, x_{3,3}$
o_4	$x_{4,3}$
o_5	$x_{5,1}, x_{5,2}$
o_6	$x_{6,2}$
o_7	$x_{7,2}$
o_8	$x_{8,1}$
o_9	$x_{9,1}, x_{9,2}$

Table 2. Binary variables associated with each set of comparison operations.

Comparison operations	Associated Binary Variables
$\{o_2\}$	$Y_{\{2\},3}, Y_{\{2\},6}, Y_{\{2\},7}, Z_{\{2\},3,2}, Z_{\{2\},3,3}, Z_{\{2\},6,2}, Z_{\{2\},7,2}$
$\{o_5\}$	$Y_{\{5\},6}, Y_{\{5\},7}, Z_{\{5\},6,2}, Z_{\{5\},7,2}$
$\{o_2, o_5\}$	$Y_{\{2,5\},6}, Y_{\{2,5\},7}, Z_{\{2,5\},6,2}, Z_{\{2,5\},7,2}$

Next, we list the constraints due to each formula as below.

Formula 1 Using operation o_2 as an example, there is exactly one binary variable is true among all the two binary variables associated with operation o_2 . Thus, we have $x_{2,2} + x_{2,3} = 1$. All the constraints due to Formula 2 are listed in the following.

$$\begin{array}{llll} x_{1,4} = 1; & x_{2,2} + x_{2,3} = 1; & x_{3,2} + x_{3,3} = 1; & x_{4,3} = 1; & x_{5,1} + x_{5,2} = 1; \\ x_{6,2} = 1; & x_{7,2} = 1; & x_{8,1} = 1; & x_{9,1} + x_{9,2} = 1. & \end{array}$$

Formula 2 Using the data dependency relation of $o_9 \rightarrow o_2$ as an example, operation o_3 can be executed if and only if operation o_9 has completed its execution. Thus, we have $x_{9,1} + 2x_{9,2} < 2x_{2,2} + 3x_{2,3}$. All the constraints due to Formula 2 are listed in the following.

$$\begin{array}{llll} x_{5,1} + 2x_{5,2} < 3x_{4,3}; & x_{8,1} < 2x_{6,2}; & x_{8,1} < 2x_{7,2}; & x_{8,1} < 2x_{3,2} + 3x_{3,3}; \\ 2x_{6,2} < 3x_{4,3}; & 2x_{7,2} < 3x_{4,3}; & 2x_{3,2} + 3x_{3,3} < 4x_{1,4}; & 3x_{4,3} < 4x_{1,4}; \\ x_{9,1} + 2x_{9,2} < 2x_{2,2} + 3x_{2,3}; & 2x_{2,2} + 3x_{2,3} < 4x_{1,4}. & & \end{array}$$

Formula 3 Suppose that the resource constraints are one multiplier, two adders, one comparator, and one multiplexer. Consider that there are three addition operations o_3 , o_7 and o_9 can be scheduled into control step 2. Since we are given two adders, we have $x_{3,2} + x_{7,2} + x_{9,2} \leq 2$. All the constraints due to Formula 3 are listed in the following.

$$\begin{array}{llll} x_{9,1} \leq 2; & x_{3,2} + x_{7,2} + x_{9,2} \leq 2; & x_{3,3} \leq 2; & x_{8,1} \leq 1; & x_{6,2} \leq 1; \\ x_{5,1} \leq 1; & x_{2,2} + x_{5,2} \leq 1; & x_{2,3} \leq 1; & x_{4,3} \leq 1; & x_{1,4} \leq 1. \end{array}$$

Formula 4 If comparison o_5 is schedule into control step 2, it is impossible to shut down operations o_6 . Thus, we have $x_{5,1} + 2x_{5,2} < 2x_{6,2} + (1 - Y_{\{5\},6}) * 4$. All the constraints due to Formula 4 are listed in the following.

$$\begin{array}{ll} 2x_{2,2} + 3x_{2,3} < 2x_{6,2} + (1 - Y_{\{2,6\}}) * 4; & 2x_{2,2} + 3x_{2,3} < 2x_{7,2} + (1 - Y_{\{2,7\}}) * 4; \\ 2x_{2,2} + 3x_{2,3} < 2x_{3,2} + 3x_{3,3} + (1 - Y_{\{2,3\}}) * 4; & x_{5,1} + 2x_{5,2} < 2x_{6,2} + (1 - Y_{\{5,6\}}) * 4; \\ x_{5,1} + 2x_{5,2} < 2x_{7,2} + (1 - Y_{\{5,7\}}) * 4. & \end{array}$$

Formula 5 Both comparison operation o_2 and comparison operation o_5 may shut down operation o_6 . Binary variable $Y_{\{2,5\},6} = 1$, if both $Y_{\{2\},6} = 1$ and $Y_{\{5\},6} = 1$. Thus, we have $Y_{\{2\},6} + Y_{\{5\},6} \leq Y_{\{2,5\},6} + 2 - 1$. All the constraints due to Formula 5 are listed in the following.

$$Y_{2,6} + Y_{5,6} \leq Y_{\{2,5\},6} + 1; \quad Y_{2,7} + Y_{5,7} \leq Y_{\{2,5\},7} + 1.$$

Formula 6 Both comparison operation o_2 and comparison operation o_5 may shut down operation o_6 . Binary variable $Y_{\{2,5\},6} = 0$, if $Y_{\{2\},6} = 0$ or $Y_{\{5\},6} = 0$. Thus, we have $Y_{\{2,5\},6} \leq Y_{\{2\},6}$ and $Y_{\{2,5\},6} \leq Y_{\{5\},6}$. All the constraints due to Formula 6 are listed in the following.

$$Y_{\{2,5\},6} \leq Y_{6,2}; \quad Y_{\{2,5\},6} \leq Y_{6,5}; \quad Y_{\{2,5\},7} \leq Y_{7,2}; \quad Y_{\{2,5\},7} \leq Y_{7,5}.$$

Formula 7 If binary variable $Y_{\{2\},6} = 1$ and binary variable $x_{2,6} = 1$, then binary variable $Z_{\{2\},6,2} = 1$. Therefore, we have $Y_{\{2\},6} + x_{2,6} \leq Z_{\{2\},6,2} + 1$. All the constraints due to Formula 7 are listed in the following.

$$\begin{aligned} Y_{2,6} + x_{6,2} &\leq Z_{2,6,2} + 1; & Y_{\{2,5\},6} + x_{6,2} &\leq Z_{\{2,5\},6,2} + 1; \\ Y_{5,6} + x_{6,2} &\leq Z_{5,6,2} + 1; & Y_{2,7} + x_{7,2} &\leq Z_{2,7,2} + 1; \\ Y_{\{2,5\},7} + x_{7,2} &\leq Z_{\{2,5\},7,2} + 1; & Y_{5,7} + x_{7,2} &\leq Z_{5,7,2} + 1; \\ Y_{2,3} + x_{3,2} &\leq Z_{2,3,2} + 1; & Y_{2,3} + x_{3,3} &\leq Z_{2,3,3} + 1. \end{aligned}$$

Formula 8 If binary variable $Y_{\{2\},6} = 0$, then binary variable $Z_{\{2\},6,2} = 0$. Therefore, we have $Z_{\{2\},6,2} \leq Y_{\{2\},6}$. All the constraints due to Formula 8 are listed in the following.

$$\begin{aligned} Z_{6,2,2} &\leq Y_{6,2}; & Z_{6,\{2,5\},2} &\leq Y_{6,\{2,5\}}; & Z_{6,5,2} &\leq Y_{6,5}; & Z_{7,2,2} &\leq Y_{7,2}; \\ Z_{7,\{2,5\},2} &\leq Y_{7,\{2,5\}}; & Z_{7,5,2} &\leq Y_{7,5}; & Z_{3,2,2} &\leq Y_{3,2}; & Z_{3,2,3} &\leq Y_{3,2}. \end{aligned}$$

Formula 9 If binary variable $x_{2,6} = 0$, then binary variable $Z_{\{2\},6,2} = 0$. Therefore, we have $Z_{\{2\},6,2} \leq x_{2,6}$. All the constraints due to Formula 9 are listed in the following.

$$\begin{aligned} Z_{2,6,2} &\leq x_{6,2}; & Z_{\{2,5\},6,2} &\leq x_{6,2}; & Z_{5,6,2} &\leq x_{6,2}; & Z_{2,7,2} &\leq x_{7,2}; \\ Z_{\{2,5\},7,2} &\leq x_{7,2}; & Z_{5,7,2} &\leq x_{7,2}; & Z_{2,3,2} &\leq x_{3,2}; & Z_{2,3,3} &\leq x_{3,3}. \end{aligned}$$

Formula 10 Consider possible situations at control step 2. If comparison operation o_2 is scheduled at control step 2 and the result of comparison operation o_2 is 0, we may shut down operation o_6 in later control steps; if comparison operation o_2 is scheduled at control step 2 and the result of comparison operation o_2 is 1, we may shut down operation o_7 in later control steps. If comparison operation o_5 is scheduled at control step 2 and the result of comparison operation o_5 is 0, we may shut down operation o_6 and operation o_7 in later control steps; if comparison operation o_5 is scheduled at control step 2 and the result of comparison operation o_5 is 1, we may shut down operation o_3 in later control steps. Therefore, if the result of comparison operation o_2 is 0 and the result of comparison operation o_5 is 0, we have $4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 20Z_{\{2\},6,2} + 20Z_{\{2,5\},6,2} - 20Z_{\{5\},6,2} - 3Z_{\{2\},7,2} \leq \text{peak_power}$; if the result of comparison operation o_2 is 0 and the result of comparison operation o_5 is 1, we have $4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 3Z_{2,7,2} + 3Z_{\{2,5\},7,2} - 3Z_{\{5\},7,2} - 20Z_{\{2\},6,2} \leq \text{peak_power}$; if the result of comparison operation o_2 is 1 and the result of comparison operation o_5 is 0, we have $4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 3Z_{\{2\},3,2} - 20Z_{\{5\},6,2} \leq \text{peak_power}$; if the result of comparison operation o_2 is 1 and the result of comparison operation o_5 is 1, we have $4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 3Z_{2,3,2} - 3Z_{\{5\},7,2} \leq \text{peak_power}$. All the constraints due to Formula 10 are listed in the following.

$$\begin{aligned} 4x_{5,1} + 20x_{8,1} + 3x_{9,1} &\leq \text{peak_power}; \\ 4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 20Z_{2,6,2} + 20Z_{\{2,5\},6,2} - 20Z_{5,6,2} - 3Z_{2,7,2} &\leq \text{peak_power}; \\ 4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 3Z_{2,7,2} + 3Z_{\{2,5\},7,2} - 3Z_{5,7,2} - 20Z_{2,6,2} &\leq \text{peak_power}; \\ 4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 3Z_{2,3,2} - 20Z_{5,6,2} &\leq \text{peak_power}; \\ 4x_{2,2} + 3x_{3,2} + 4x_{5,2} + 20x_{6,2} + 3x_{7,2} + 3x_{9,2} - 3Z_{2,3,2} - 3Z_{5,7,2} &\leq \text{peak_power}; \\ 4x_{2,3} + 3x_{3,3} + x_{4,3} &\leq \text{peak_power}; \end{aligned}$$

$$\begin{aligned}
4x_{2,3} + 3x_{3,3} + x_{4,3} &\leq \text{peak_power}; \\
4x_{2,3} + 3x_{3,3} + x_{4,3} - 3Z_{2,3,3} &\leq \text{peak_power}; \\
4x_{2,3} + 3x_{3,3} + x_{4,3} - 3Z_{2,3,3} &\leq \text{peak_power}; \\
x_{1,4} &\leq \text{peak_power}.
\end{aligned}$$

After solving these ILP formulations, we have that $x_{1,4} = x_{2,3} = x_{3,3} = x_{4,3} = x_{5,1} = x_{6,2} = x_{7,2} = x_{8,1} = x_{9,2} = Y_{\{5\},6} = Y_{\{5\},7} = Z_{\{5\},6,2} = Z_{\{5\},7,2} = 1$, and the values of other binary variables are 0. As a result, we obtain the scheduled CDFG as shown in Fig. 5. The peak power obtained by our ILP approach is only 24, which is also the lower bound of the peak power.

5. HEURISTIC ALGORITHM

In this section, we present a heuristic algorithm to deal with the same problem. For the convenience of presentation, we define a term ‘‘hottest control step’’ as below. For a scheduled CDFG, a control step is called the hottest control step, if the peak power (among all the control steps) happens at this control step.

```

Procedure Heuristic_Algorithm()
begin
use list scheduling to derive an initial scheduled CDFG S;
unmark all the operations;
while (1)
  begin
  obtain a new scheduled CDFG S' by moving unmarked operations in the hottest
  control step of S;
  mark all the moved operations;
  obtain a new scheduled CDFG S'' by adding as many soft edges into S' as possible;
  if (the peak power of S == the peak power of S'')
    then return(S);
    else S = S'';
  end
end.

```

Fig. 7. Our heuristic algorithm.

Fig. 7 gives the pseudo code of our heuristic algorithm. We use list scheduling [2] to derive an initial scheduled CDFG. Then, we enter an iteration process of while-loops. Each while-loop has two steps: at the first step, we reduce the peak power by moving operations in the hottest control step to other control steps; at the second step, we reduce the peak power by adding as many soft edges as possible. The iteration process repeats until the peak power cannot be further reduced.

In the following, we discuss the method of moving operations. At the beginning, we unmark each operation. An operation becomes marked after it is moved. Note that only unmarked operations can be moved; in other words, each operation can be moved at most

one time. In each while-loop, we examine all the unmarked operations in the hottest control step in the decreasing order of their power consumptions. We cannot move an unmarked operation to another control step, unless the following two rules are satisfied at the same time:

- (1) the design constraints (*i.e.*, timing and resource constraints) are met; and
- (2) the resulting peak power (among all the control steps) is less than or equal to the original peak power (among all the control steps).

If an unmarked operation can be moved to more than one control steps, it is moved to the control step that leads to the smallest peak power (among all the control steps).

We analyze the time complexity of our heuristic algorithm as below. In each while-loop, the time complexity to move operations in the hottest control step is $O(n * t)$, where n is the number of operations and t is the number of control steps. The iteration process of while-loops repeats until no operation can be moved. Since each operation can be moved at most one time, the number of iteration process of while-loops is at most n . Therefore, the time complexity of our heuristic algorithm is $O(n^2 * t)$.

Take the CDFG shown in Fig. 1 (b) for illustration. Suppose that the design constraints (*i.e.*, timing and resource constraints) are four control steps, one multiplier, two adders, one comparator, and one multiplexer. After list scheduling is applied, we obtain the scheduled CDFG as shown in Fig. 2. The power consumption at control step 1, control step 2, control step 3, and control step 4 are 27, 31, 1, and 1, respectively. Therefore, the peak power is 31. Then, we unmark each operation and enter the iteration process of while-loops.

At the first iteration of the while-loop, we find that control step 2 is the hottest control step in the scheduled CDFG shown in Fig. 2. Therefore, we attempt to move operations in control step 2 to other control steps. We find that, in control step 2, operation o_2 , operation o_3 , operation o_6 , and operation o_7 are unmarked. We examine these unmarked operations according to the decreasing order of power consumptions (*i.e.*, in the sequence of operation o_6 , operation o_7 , operation o_3 , and operation o_2). Due to the design constraints, operation o_6 and operation o_7 cannot be moved to other control steps. On the other hand, we find that operation o_3 can be moved to control step 3. Fig. 8 gives the obtained scheduled CDFG. The power consumption at control step 1, control step 2, control step 3, and control step 4 become 27, 27, 5, and 1, respectively. Therefore, the peak power is reduced from 31 to 27. Next, we also find that operation o_2 can be moved to control step 3. Fig. 9 gives the obtained scheduled CDFG. The power consumption at control step 1, control step 2, control step 3, and control step 4 become 27, 24, 8, and 1, respectively. The power consumption at control step 2 is reduced; however, the peak power is still 27. After that, we have examined all the unmarked operations in control step 2. Since operation o_2 and operation o_3 are moved, the two operations become marked. Then, we attempt to add as many soft edges as possible. Two soft edges can be added: one soft edge is from comparison operation o_5 to multiplication operation o_6 , and one soft edge is from comparison operation o_5 to addition operation o_7 . Fig. 10 gives the obtained scheduled CDFG. Therefore, when the result of comparison operation o_5 is 0, the power consumption at control step 1, control step 2, control step 3, and control step 4 are 27, 4, 8, and 1, respectively; when the result of comparison operation o_5 is 1, the power

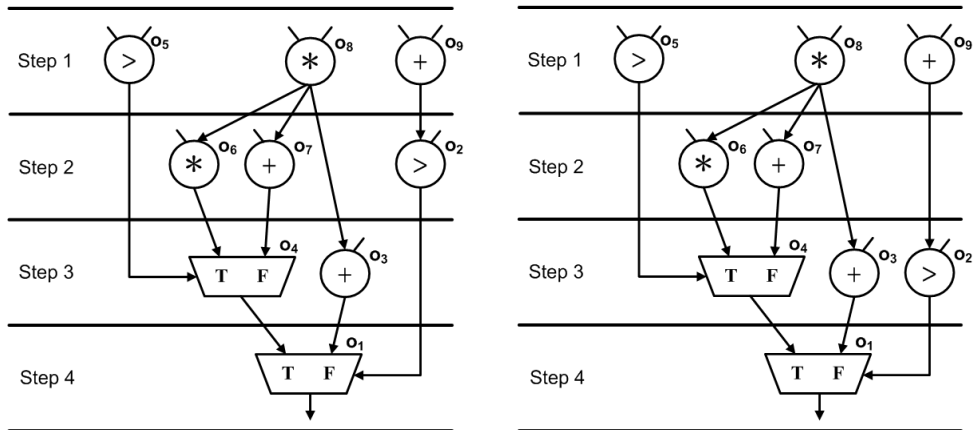


Fig. 8. Scheduled CDFG after operation o_3 is moved. Fig. 9. Scheduled CDFG after operation o_2 is moved.

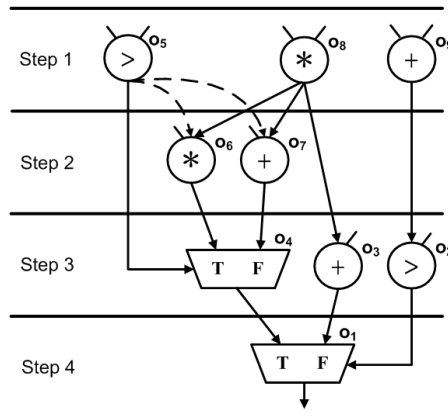


Fig. 10. Scheduled CDFG after soft edges are added.

consumption at control step 1, control step 2, control step 3, and control step 4 are 27, 20, 8, and 1, respectively. The power consumption at control step 2 is further reduced; however, the peak power is still 27. Thus, after the first iteration of the while-loop is finished, the peak power is 27.

At the second iteration of the while-loop, we find that control step 1 is the hottest control step in the scheduled CDFG shown in Fig. 10. Therefore, we attempt to move unmarked operations in control step 1 to other control steps. We find that, in control step 1, operation o_8 , operation o_5 , and operation o_9 are unmarked. We examine these unmarked operations according to the decreasing order of power consumptions (*i.e.*, in the sequence of operation o_8 , operation o_5 , and operation o_9). Due to the design constraints, we find that operation o_8 and operation o_5 cannot be moved to other control steps.² On the other hand, we find that operation o_9 can be moved to control step 2. Fig. 5 gives the obtained scheduled CDFG. As a result, when the result of comparison operation o_5 is 0,

² Operation o_5 cannot be moved to other control steps, because the control dependency constraints imposed by soft edges.

the power consumption at control step 1, control step 2, control step 3, and control step 4 are 23, 8, 8, and 1, respectively; when the result of comparison operation o_5 is 1, the power consumption at control step 1, control step 2, control step 3, and control step 4 are 23, 24, 8, and 1, respectively. Therefore, the peak power becomes 24. After that, we have examined all the operations in control step 1. Since operation o_9 is moved, it becomes marked. Then, we attempt to add as many soft edges as possible. But no soft edge can be added. Thus, after the second iteration of the while-loop is finished, the peak power is 24.

At the third iteration of the while-loop, we find that control step 2 is the hottest control step in the scheduled CDFG shown in Fig. 5. However, all the operations in control step 2 have been marked. No operation can be moved and no soft edges can be added. As a result, the peak power cannot be further reduced. Therefore, our heuristic algorithm returns the scheduled CDFG as shown in Fig. 5.

In this example, our heuristic algorithm derives the same scheduled CDFG as our ILP approach. In other words, in this example, our heuristic algorithm has achieved the lower bound of peak power. However, it should be pointed out that: solving the ILP formulation (that guarantees achieving the lower bound of peak power) is an NP-hard problem. In fact, our heuristic algorithm does not guarantee achieving the lower bound of peak power.

6. EXPERIMENTAL RESULTS

Nine benchmark circuits are used to test the effectiveness of our approach. Benchmark circuits GCD [16], Jian [17], Mult [18], G2 [19], G5 [20] are popular DSP applications and widely used in the high-level synthesis community; benchmark circuits Dist1 and Dist2 are the representative functions adopted from the MediaBench suite [21]; and benchmark circuits R1 and R2 are data-dominated applications adopted from [13]. Table 3 describes the characteristics of benchmark circuits. The column *Operations* describes the numbers of different types of operations. The column * denotes the number of multiplication operations. The column + denotes the number of addition operations. The column – denotes the number of subtraction operations. The column > denotes the number of comparison operations. The column # denotes the number of multiplexers. The column

Table 3. Characteristics of benchmark circuits.

Circuit	Operations					Percentage of Conditionals
	*	+	–	>	#	
GCD	0	2	0	3	3	75.0%
Jian	0	10	0	3	3	37.5%
Mult	0	7	3	2	2	28.6%
G2	9	9	0	3	3	25.0%
G5	0	16	8	2	2	14.3%
Dist1	0	48	48	16	16	25.0%
Dist2	64	192	64	3	3	1.8%
R1	15	6	6	27	28	67.1%
R2	26	4	4	29	45	68.5%

Percentage of Conditionals gives the percentage of comparisons operations and multiplexers in each benchmark circuit. For instance, in benchmark circuit GCD, the percentage of comparisons operations and multiplexers is $(3 + 3)/(2 + 3 + 3) = 75.0\%$.

Our approach is implemented on the Window-XP platform with P4-2.4GHz CPU and 512M Bytes RAM. We use Extended Lingo Release 8.0 as the ILP solver. We also use C++ programming language to implement our heuristic algorithm. Note that our approach is the first work that deals with the problem of simultaneous operation scheduling and power management for peak power minimization. For the purpose of comparisons, we also implement the two-step process: at the first step, we use Shiue's work [15] as the operation scheduling algorithm³; at the second step, we add as many soft edges as possible.

In our experiments, we adopt the functional unit library used in [10]: the power consumptions of multiplier, ALU, comparator, and multiplexer are assumed to be 20, 4, 3, and 1, respectively. In addition, we assume that each functional unit takes one control step to complete an operation. Table 4 gives our experimental results, including the two-step process, our ILP approach, and our heuristic algorithm. It is noteworthy to mention that our ILP approach guarantees minimizing the peak power. The column Resources describes the given resource constraint (#mul, #alu, #comp, #mux), where #mul, #alu, #comp, and #mux denote the number of multipliers, the number of ALUs, the number of comparators, and the number of multiplexers, respectively. For example, (1, 6, 3, 3) means that one multiplier, six ALUs, three comparators, and three multiplexers. The column steps denotes the number of control steps. The column Reduction denotes the

Table 4. Comparisons on peak powers.

Circuit	Constraints		Peak Power			Reduction (%)	
	Resources	Steps	Two-Step Process	Our ILP	Our Heuristic	Our ILP	Our Heuristic
GCD	(0, 2, 1, 1)	5	7	5	5	28.57%	28.57%
Jian	(0, 3, 1, 1)	7	9	7	8	22.22%	11.11%
		8	7	6	6	14.29%	14.29%
Mult	(0, 3, 1, 1)	6	9	7	7	22.22%	22.22%
G2	(2, 2, 1, 1)	8	46	40	43	13.04%	6.52%
		9	43	23	23	46.51%	46.51%
G5	(0, 4, 2, 2)	9	12	9	9	25.00%	25.00%
		10	9	7	7	22.22%	22.22%
Dist1	(0, 4, 3, 3)	38	13	11	12	15.38%	7.69%
		39	10	8	8	20.00%	20.00%
Dist2	(1, 6, 3, 3)	100	27	20	26	25.93%	3.70%
		105	26	20	20	23.08%	23.08%
R1	(2, 5, 3, 3)	11	52	46	49	11.54%	5.77%
		15	43	32	35	25.58%	18.60%
R2	(2, 5, 3, 3)	15	56	49	53	12.50%	5.36%
		20	53	40	49	24.53%	7.55%

³ Note that Shiue's work [15] is the only paper that studies the problem of operation scheduling for peak power minimization. Therefore, we use Shiue's work [15] as the operation scheduling algorithm of the two-step process.

Table 5. Comparisons on CPU times.

Circuit	Constraints		CPU Time (sec)		
	Resources	Steps	Two-Step Process	Our ILP	Our Heuristic
GCD	(0, 2, 1, 1)	5	1	< 1	1
Jian	(0, 3, 1, 1)	7	< 1	21	< 1
		8	< 1	37	< 1
Mult	(0, 3, 1, 1)	6	< 1	< 1	< 1
G2	(2, 2, 1, 1)	8	< 1	3	< 1
		9	1	12	1
G5	(0, 4, 2, 2)	9	< 1	32	< 1
		10	< 1	16	< 1
Dist1	(0, 4, 3, 3)	38	90	153	31
		39	93	161	35
Dist2	(1, 6, 3, 3)	100	169	215	51
		105	180	223	58
R1	(2, 5, 3, 3)	11	20	76	3
		15	23	80	3
R2	(2, 5, 3, 3)	15	25	98	5
		20	26	102	6

relative reduction of our approach, including our ILP approach and our heuristic algorithm, over the two-step process. The relative reduction of our approach over the two-step process is obtained by (the peak power of the two-step process – the peak power our approach)/(the peak power of two-step process). From Table 4, we find that: the average peak power reduction of our ILP approach is 22.04%, and the average peak power reduction of our heuristic algorithm is 16.76%.

Table 5 gives the comparisons on the CPU times of different approaches, including the two-step process, our ILP approach, and our heuristic algorithm. Note that the operation scheduling algorithm of the two-step process, *i.e.*, Siue's work [15], is also an ILP approach. Therefore, in each benchmark circuit, the CPU time of the two-step process is even larger than the CPU time our heuristic algorithm. With an analysis to Table 5, we find that: in all the benchmark circuits, the CPU times of the two-step process and the CPU times of our ILP approach are at most few minutes, and the CPU times of our heuristic algorithm are less than one minute.

7. CONCLUSIONS

In this paper, we study the simultaneous application of operation scheduling and power management for peak power minimization. To the best of our knowledge, our paper is the first work to deal with this problem. Our contribution includes the following two aspects. First, we propose an integer linear program to formally formulate this problem. It is noteworthy to mention that our ILP approach guarantees minimizing the peak power. Second, we also propose a heuristic algorithm to solve this problem in polynomial time complexity. Benchmark data show that our heuristic algorithm can achieve near-optimal solutions within small CPU times.

REFERENCES

1. D. D. Gajski, N. D. Dutt, and B. M. Pangrle, "Silicon compilation (tutorial)," in *Proceedings of International Conference on Custom Integrated Circuits*, 1986, pp. 102-110.
2. S. Davidson, D. Landskov, B. D. Shriver, and P. W. Mallett, "Some experiments in local microcode compaction for horizontal machines," *IEEE Transactions on Computers*, Vol. C-30, 1981, pp. 460-477.
3. P. Paulin and J. Knight, "Force directed scheduling for behavioral synthesis of ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 8, 1989, pp. 661-679.
4. C. T. Hwang, J. H. Lee, and Y. C. Hsu, "A formal approach to the scheduling problem in high level synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, 1991, pp. 464-475.
5. R. Camposano, "Path-based scheduling for synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, 1991, pp. 85-93.
6. T. Kim, J. W. S. Liu, and C. L. Liu, "A scheduling algorithm for conditional resource sharing," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, 1991, pp. 84-87.
7. J. Monterio, S. Devadas, P. Ashar, and A. Mauskar, "Scheduling technique to enable power management," in *Proceedings of IEEE/ACM International Conference on Design Automation*, 1996, pp. 349-352.
8. P. Prabhakaran and P. Banerjee, "Parallel algorithms for force directed scheduling of flattened and hierarchical signal flow graphs," *IEEE Transactions on Computers*, Vol. 48, 1999, pp. 762-768.
9. P. Faraboschi, J. A. Fisher, and C. Young, "Instruction scheduling for instruction level parallel processors," *Proceedings of the IEEE*, Vol. 89, 2001, pp. 1638-1659.
10. C. Chen and M. Sarrafzadeh, "Power-manageable scheduling technique for control dominated high-level synthesis," in *Proceedings of IEEE Design, Automation and Test in Europe Conference and Exhibition*, 2002, pp. 1016-1020.
11. S. O. Memik, A. Srivastava, E. Kursun, and M. Sarrafzadeh, "Algorithmic aspects of uncertainty driven scheduling," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Vol. 3, 2002, pp. 763-766.
12. S. H. Huang, C. H. Chiang, and C. H. Cheng, "Three-dimension scheduling under multi-cycle interconnect communications," *IEICE Electronics Express*, Vol. 2, 2005, pp. 108-114.
13. S. H. Huang, C. H. Cheng, C. H. Chiang, and C. M. Chang, "An ILP approach to the simultaneous application of operation scheduling and power management," in *Proceedings of Joint Conference on Information Sciences*, 2006, pp. 735-738.
14. S. H. Huang and C. H. Cheng, "An ILP approach to the slack driven scheduling problem," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, Vol. 89-A, 2006, pp. 1852-1858.
15. W. T. Shiue, "High level synthesis for peak power minimization using ILP," in *Proceedings of IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, 2000, pp. 103-112.
16. F. F. Hsu, E. M. Rudnick, and J. H. Patel, "Testability insertion in behavioral de-

- scriptions,” in *Proceedings of IEEE International Symposium on System Synthesis*, 1996, pp. 139-144.
17. J. Li and R. K. Gupta, “An algorithm to determine mutually exclusive operations in behavioral descriptions,” in *Proceedings of IEEE/ACM Design Automation and Test in Europe*, 1998, pp. 457-465.
 18. K. Wakabayashi and H. Tanaka, “Global scheduling independent of control dependencies based on condition vectors,” in *Proceedings of IEEE/ACM Design Automation Conference*, 1992, pp. 112-115.
 19. C. J. Tseng, R. W. Wei, S. G. Rothweiler, M. Tong, and A. K. Bose, “Bridge: a versatile behavioral synthesis system,” in *Proceedings of IEEE/ACM Design Automation Conference*, 1988, pp. 415-420.
 20. T. Kim, J. W. S. Liu, and C. L. Liu, “A scheduling algorithm for conditional resource sharing,” in *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, 1991, pp. 84-87.
 21. C. Lee, M. Potkonjak, and W. H. Maggione-Smith, “MediaBench: a tool for evaluating and synthesizing multimedia and communications systems,” in *Proceedings of IEEE International Symposium on Microarchitecture*, 1997, pp. 330-335.
 22. C. Gopalakrishnan and S. Katkooori, “Resource allocation and binding approach for low leakage power,” in *Proceedings of IEEE International Symposium on VLSI Design*, 2003, pp. 297-302.
 23. F. J. Kurdahi and A. C. Parker, “REAL: a program for register allocation,” in *Proceedings of IEEE/ACM Design Automation Conference*, 1987, pp. 210-215.
 24. C. Tseng and D. P. Siewiorek, “Automatic synthesis of data paths in digital systems,” *IEEE Transactions on Computer-Aided Design*, Vol. 5, 1986, pp. 379-395.

APPENDIX

We can use the power gating technique [22] to shut down the functional units that execute unused operations. In this Appendix, we study the implementation of control logics (*i.e.*, the sleep signals of functional units). Note that the control logics cannot be determined until operations are scheduled and assigned to functional units. In this paper, we do not discuss the problem of assigning operations to functional units. By using our scheduled CDFG, the designer can directly apply existing algorithms, such as left edge approach [23] or minimum clique partitioning [24], to assign operations to functional units. If the readers are interested in the problem of assigning operations to functional units, please refer to [23, 24].

Take the CDFG shown in Fig. 1 (b) for illustration. Suppose that the design constraints (*i.e.*, timing and resource constraints) are four control steps, one multiplier, two adders (called add1 and add2, respectively), one comparator, and one multiplexer. By applying our ILP approach, we obtained a scheduled CDFG as shown in Fig. 5. Next, we assign operations to functional units. Without loss of generality, here, we assume that: addition operation o_3 and addition operation o_7 are assigned to adder add1, and addition operation o_9 is assigned to adder add2. Assume that a functional unit is shut down when its sleep signal is 1. We discuss the control logics as below.

First, we do not consider soft edges. We have the following control logics:

- (1) The multiplier. At control step 1 and control step 2, the sleep signal is 0; at control step 3 and control step 4, the sleep signal is 1.
- (2) The adder add1. At control step 2 and control step 3, the sleep signal is 0; at control step 1 and control step 4, the sleep signal is 1.
- (3) The adder add2. At control step 2, the sleep signal is 0; at control step 1, control step 3, and control step 4, the sleep signal is 1.
- (4) The comparator. At control step 1 and control step 3, the sleep signal is 0; at control step 2 and control step 4, the sleep signal is 1.
- (5) The multiplexer. At control step 3 and control step 4, the sleep signal is 0; at control step 1 and control step 2, the sleep signal is 1.

Next, we discuss extra control logics (due to soft edges). There are two soft edges: a soft edge from comparison operation o_5 to multiplication operation o_6 , and a soft edge from comparison operation o_5 to addition operation o_7 . Due to the two soft edges, the control logics have the following modifications (for shutting down unused operations).

- (1) The multiplier. At control step 2, if the result of comparison operation o_5 is 0, the sleep signal is 1.
- (2) The adder add1. At control step 2, if the result of comparison operation o_5 is 1, the sleep signal is 1.

From this example, we also know that: extra control logics (due to soft edges) are rather simple and small. Therefore, the power consumptions of extra control logics are ignored in previous works [7, 10, 13]. Following the same assumption in previous works [7, 10, 13], this paper also ignores the power consumptions of extra control logics (for shutting down unused operations).



Shih-Hsu Huang (黃世旭) received the B.S. degree in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1989, the M.S. degree in Computer Science from National Tsing Hua University, Hsinchu, in 1991, and the Ph.D. degree in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1995. From 1995 to 2000, he was with Computer and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, rising to the position of deputy manager of IC design department, responsible for the design of high performance IC's. In 2000, he joined the department of Electronic Engineering, Chung Yuan Christian University, Chungli, Taiwan, as a faculty member, where he is currently an Associate Professor. Dr. Huang co-received the Most Popular Paper Award from the 18th VLSI Design/CAD Symposium, Taiwan, in 2007. His research interests include high-level synthesis, timing optimization, and physical design.



Chun-Hua Cheng (程駿華) received the B.S. degree in Electronic Engineering from Chun Yuan Christian University, Chungli, Taiwan, R.O.C., in 2003, and the M.S. degree in Electronic Engineering from Chung Yuan Christian University, Chungli, Taiwan, in 2005. He is presently working toward the Ph.D. degree in Electronic Engineering at Chung Yuan Christian University, Chungli, Taiwan. Mr. Cheng co-received the Most Popular Paper Award from the 18th VLSI Design/CAD Symposium, Taiwan, in 2007. His research interests include timing optimization and high-level synthesis.