A Minimal Power Scheduling Algorithm for Low Power Circuit Design

Chi-Ho Lin

Department of Computer Science, Semyung University
San 21-1, Shinwol-dong Chechon, Chungbuk, 390-711, Korea
TEL : +82-43-649-1272, FAX : +82-43-644-2111
E-mail :ich410@semyung.ac.kr

Abstract

In this paper, we present an intermediate representation CDFG(Control Data Flow Graph) and an efficient scheduling technique for low power circuit design. The proposed CDFG represents control flow, data dependency and such constraints as resource constraints and timing constraints. In the scheduling technique, the constraints are substituted by subgraphs, and then the number of subgraphs is minimized by using the inclusion and overlap relation efficiently. Also, iterative rescheduling process are performed in a minimum bound estimation, starting with the as soon as possible as scheduling result , so as to reduce the power consumption in low power design. The effectiveness of the proposed algorithm has been proven by the experiment with the benchmark examples.

1. Introduction

The objectives of high level synthesis is to translate a behavioral description specified in a high level language into an efficient register-transfer level structure that implements the behavior. In the high level synthesis, the behavioral description is usually transformed into a CDFG(Control Data Flow Graph) as an intermediate representation. A typical high level synthesis process involves several subtasks including behavioral transformations, module selection, clock period selection, scheduling, and resource sharing, and RTL circuit generation[1].

High level synthesis has a large impact on power consumption, which, if properly exploited, can lead to large power savings. Recent work has shown that the most savings in power consumption are often obtained at the higher levels of the design hierarchy[2-4].

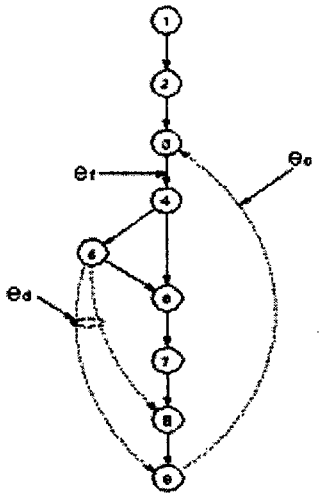
Scheduling is one of the most important tasks in high level synthesis. It determines the cycle-by-cycle behavior of a design by assigning parts of the computation to be performed to particular clock cycles(control steps or control states)[5-9]. In this paper, we concentrate on reducing power management into scheduling algorithm used in high level synthesis comprises of the sequence of steps by means of which an algorithmic specification is translated into hardware. These steps involve breaking down the algorithm into primitive operations, and associating each operation with the time interval in which it will be executed (called operation scheduling). In this paper, we present a new VHDL intermediate representation CDFG and an efficient scheduling technique for low power design. In the proposed scheduling algorithm, the constraints are substituted by subgraphs, and then the number of subgraphs is minimized by using the inclusion and overlap relation efficiently. Also, iterative rescheduling process are performed in a minimum bound, starting with the as soon as possible as scheduling result , so as to reduce the power consumption in low power design. These methods speed up our algorithm considerably without loss of optimally in the scheduling result. The rest of the paper is organized as follows. Section 2 describes the VHDL intermediate representation.. Section 3 presents an algorithm for the efficient low power scheduling technique. Section 4 discuss an in a minimum bound rescheduling process. Section 5 describes experimental result in our proposed algorithm, and finally section 6 gives conclusion.

2. The VHDL intermediate representation

The VHDL analyzer translates the behavioral specification with VHDL description into the internal data structure suitable for the application areas of high level synthesis system.

```
entity example is
Port (
 branch_pc, ibus: in BIT_VECTOR(3 downto 0);
       branch, ire: in BIT;
    ire, ppc, obus:  out BIT_VECTOR(3 downto 0);
);
end example;
architecture behavior of example is
begin
  process
  variable pc, oldpc: BIT_VECTOR(3 downto 0);
    begin
    ppc <= pc;                                  (1)
    popc <= oldpc;                              (2)
    obus <= ibus + "0100";                      (3)
    if (branch = '1') then                      (4)
     pc := branchpc;                            (5)
    end if;                                     (6)
    wait until (ire = '1')                      (7)
    oldpc := pc;                                (8)
    pc := pc + "0100";                          (9)
    assert (NOW - pc' event <= 200 ns)
      report "Max. Time violation";
  end process;
end behavior;
```

(a)



(b)

Fig 1. VHDL description and CDFG.
(a) VHDL description.
(b) CDFG.

The internal data structure, CDFG, which represents both the control flow and data flow effectively, is constructed. The CDFG represents the constraints which limit the hardware design such as conditional branch, sequential operation and time constraints.

Fig. 1(a) shows a VHDL description for the example. Fig. 1(b) shows a CDFG for the example of Fig. 1(a). The semantics of VHDL description can be defined by the internal form generated by the VHDL.

Let D = (V, E) be a VHDL description, V the set of nodes for VHDL statements that represent VHDL statements in D and E directed edges that appear relations between nodes in D.

The internal representation CDFG of D is given by

$$CDFG = (G_f, G_c, G_t)$$

(Definition 1) Sequential graph $G_f = (V, E_f)$ consists of V the set of nodes for VHDL statements and directed edges $E_f = \{ (v_1, v_2) \mid v_1, v_2 \subset V, v_1$ is the predecessor of $v_2 \}$.

(Definition 2) Hardware constraint graph $G_c = (V, E_c)$ consists of V the set of nodes for VHDL statements and directed edges $E_c = \{ (v, w) \mid v, w \subset V$, there is one among 4 hardware constraints starting with v and ending with w $\}$.

(Definition 3) Timing relation graph $G_t = (V, E_t)$ consists of V the set of nodes for VHDL statements and directed edges $E_t = \{ (v_1, v_2) \mid v_1, v_2 \subset V$, there is a timing constraint starting with v_1 and ending with $v_2 \}$.

Notice that Sequential graph $G_f = (V, E_f)$ does not exist for clock parts and waiting part. CDFG is a control flow graph which represents conditional branches and loops efficiently. Also it represents data dependency and constraints such as hardware resource and timing.

3. The efficient low power scheduling technique

In order to represent control flow, data dependency and such constraints as resource constraints and timing constraints effectively, the CDFG represents the constraints which limit the hardware design in such a way :

1) no variable is assigned more than once in each control step
2) no I/O port is accessed more than once in each control step
3) the total delay of operations in each control step is not greater than the given control step-length
4) all designer imposed constraints for scheduling particular operations in different control steps are satisfied.

In order to satisfy any of the above conditions, the proposed scheduling algorithm generates constraints between two nodes that must be scheduled into different control steps. In the proposed scheduling algorithm, the constraints are substituted by subgraphs, and then the number of subgraphs (that is the number of the constraints) is minimized by using the inclusion and overlap relation among subgraphs.

3.1 The removal of subgraph with inclusion relation

The hardware constraints are substituted by subgraphs, and the number of the constraints is minimized by using the inclusion and overlap relation among subgraphs. The subgraph minimization for the hardware constraints optimizes the number of control steps needed to execute the nodes in the CDFG.

(Definition 4) Regardless of conditional branches, if both edges E_c is in inclusion relation, an included edge can be removed .

The subgraph minimization for the hardware constraints optimizes the number of control steps needed to execute the nodes in the CDFG. In an example of Fig 2., if edge(2,9) include both edge(3,4) and edge(2,9), An edge(2,9) is removed. Just, because the nodes 3 and 4 cannot operate in the same control step, constraints indicated by the nodes 2 and 9 can be removed.

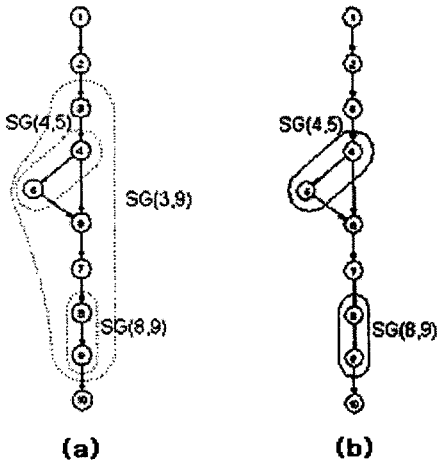
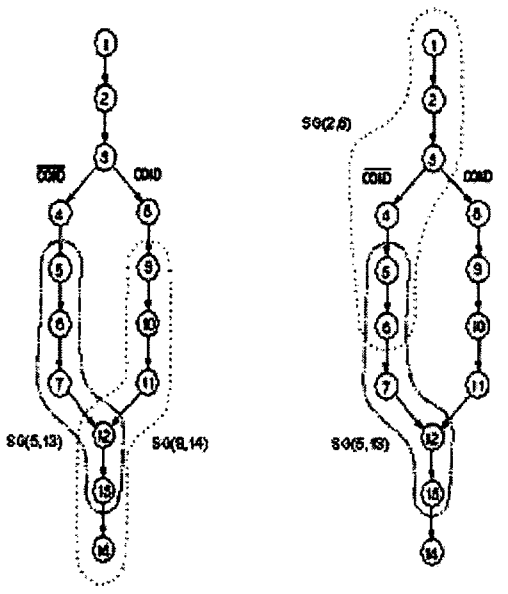


Fig 2. The removal of subgraph with inclusion relation
(a) A subgraph with inclusion relation
(b) The removal subgraph of Fig (a).

3.2 The modification of subgraph with overlap relation

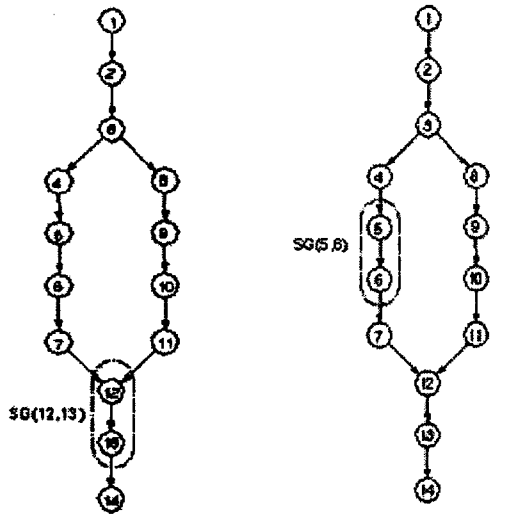
After the removal of subgraph with inclusion relation, we have to search for subgraphs with overlap relation to minimize total operation time in replacing many subgraphs with new subgraph. Accordingly to be scheduled as soon as possible and to minimize total operation time for the nodes in the CDFG, the overlap subgraphs must be minimized with the following priority.

As shown in Fig 3. and Fig 4., if there is an overlap relation between edges E_c in subgraph, Overlap part is replaced by new edge E_c ,and then old new edge E_c is removed repeatedly until not exist overlap relation in the CDFG.



(a) (b)

Fig. 3. Overlap subgraphs with a conditional branch



(a) (b)

Fig. 4. The modified results of Fig. 3.

We then partition each path in the CDFG into control steps in such a way that our scheduling algorithm to satisfy any of the conditions, determines control steps in an expression (1)

T(N) = The weighted value for adjacent edges + MAX(operation time in predecessor nodes) (1)

4. A minimum bound rescheduling process

The basic behind minimum bound rescheduling process stems from the pigeon hole principle: if N operations are scheduled over K control steps, then it is guaranteed that at least $\lceil N/K \rceil$ operations are scheduled into some control step among those K control steps. This can be stated slightly differently we are talking about a control steps interval $Z=[X, Y]$ whose length is $Y-X+1$. In this case, if N operations of type T are scheduled in the interval Z, then at least $\lceil N/(Y-X+1) \rceil$ FUs of type T are required. Now given a particular operation O_i, then clearly O_i, is guaranteed to be scheduled in Z if $[ASAP_i, ALAP_i] \subseteq Z$.

For each interval Z, we find the number of operations of type T guaranteed to be scheduled during that interval, and estimate the lower bound on the number of FUs of type T.

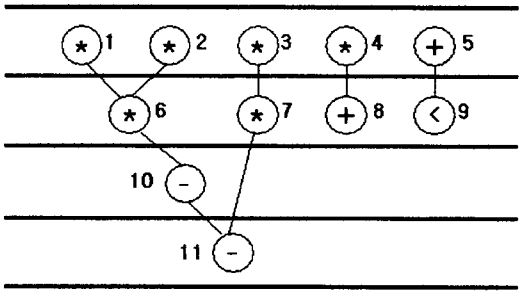


Fig. 5. An example of scheduled result

For example in Fig 5., the four multipliers O_1, O_2, O_3, and O_6 are guaranteed to be executed in control step interval [1,2], since their time frames are fully included in this interval example as shown in Fig 6. So [4/2]=2 is estimated as a candidate of the number of multipliers for this interval. To get a tighter minimum bound, these candidates are estimated over all the control step intervals and the maximum one is selected. In this case, MAX(0,1,2)=2 is finally chosen as the minimum bound on the number of multipliers. In a similar way, the minimum bound on ALU count is estimated as 2

The initial time frame of O_3 is [1,2] as shown in Fig 6. However, if we assume that the number of multipliers available is equal to the low bound on the number on the number of multipliers(=2), O_3 cannot be scheduled into

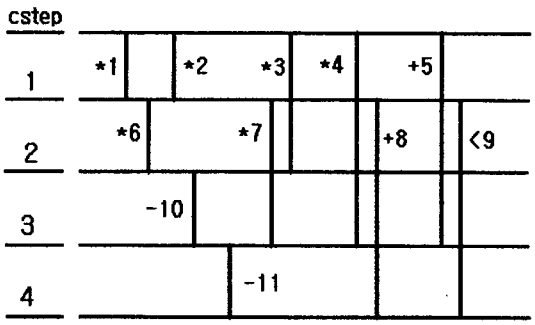


Fig. 6. Time frames of operations

control step 1, since at least 2 other multipliers O_1 and O_2 are guaranteed to be scheduled into control step. Therefore, the time frame of O_3 shrinks to [2,2]. In similar way, we can adjust the time frames of the operations as shown in Fig. 7.

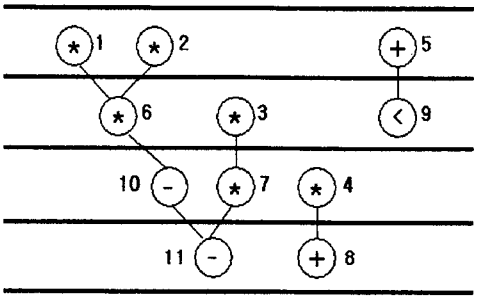


Fig. 7. A minimum bound rescheduled result

5. Experimental Results

In this paper, our scheduling algorithm have been implemented in C++ and UltraSPARC III system. Also, it have been tested on the Table 1. In this experiment results, we results the calculated power reduction ratio to adopt the HLS benchmark through the result of an optimal data path scheduling technique for low power circuit design.

Our methods were implemented within the framework of low power high level synthesis. The effectiveness of the proposed algorithm has been proven by the experimental result for benchmarks. Especially FIR system power consumption is reduced 38%.

Table 1. The low power synthesis result for benchmarks.

	Circuit	#Csteps	#Operations	#Registers	Power Red(%)
[9]	Diffeq	4	4	7	
	Tseng	4	5	6	
	FIR	10	5	24	
	EWF	13	5	11	
OURS	Diffeq	4	4	4	27
	Tseng	4	5	4	26
	FIR	10	5	10	38
	EWF	13	5	13	12

6. Conclusion

We presented a new VHDL intermediate representation CDFG and an efficient scheduling technique for low power circuit design.

We have presented a scheduling algorithm which, for a given throughput, exploits the slack available to operations to obtain a schedule that power management technique. This more constrained scheduling process may lead to a large number of execution units required.

The scheduling technique for minimizing the operation time and handling the conditional branch effectively for ASIC design have been performed. Unlike most previous work, we also consider the interaction among these tasks in order to better explore the design space. We have implemented the algorithm, and presented experimental results to demonstrate its effectiveness. Also, we have obtained a solution that maximizes the ability to do power management while still meeting user specified throughput and hardware resource constraints.

Reference

[1] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, ``Low power CMOS digital design,'' *IEEE J. Solid State Circuits*, pp. 473--484, Apr. 1992.

[2] A. P. Chandrakasanet al., ``Optimizing power using transformations, *"IEEE Trans. Computer Aided Design*, vol. 14, pp. 12-31, Jan. 1995.

[3] R. Mehra and J. Rabaey, ``Behavioral level power estimation and exploration,'' *in Proc. Int. Wkshp. Low Power Design*, pp. 197-202, Apr. 1994.

[4] L. Goodby, A. Orailoglu, and P. M. Chau, ``Microarchitectural synthesis of performance constrained, low power VLSI designs,'' *in Proc. Int. Conf. Computer Design*, pp. 323--326, Oct. 1994.

[5] A. Ghosh, " Estimation of Average Switching Activity in Combination and Sequential Circuits", *in Proc. 29th DAC*, June 1992, pp.253-259

[6] P. Landman, " Power Estimation of High-Level Synthesis", *in Proc. European DAC*, Feb. 1993, pp.361-366

[7] A. Chandarksan et al., "HYPER-LP: A System fo Power Minimization Using Architecture Transformation," in Proc. ICCAD, Nov. 1992, pp.300-303

[8] R. Martin, "Power-Profiler : Optimizing ASICs Power Consumption at the Behavioral Level", in Proc. 32nd DAC, June 1995, pp.42-47

[9] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design Circuits and Systems*, Kluwer Academic Publishers, 1995.