Power-Manageable Scheduling Technique for Control Dominated High-Level Synthesis

Chunhong Chen
Dept. of Electrical and Computer Eng.
The University of Windsor, Ontario, CANADA

Majid Sarrafzadeh
Computer Science Department
UCLA, Los Angeles, USA

Abstract

Optimizing power consumption at high-level is a critical step towards power-efficient digital system designs. This paper addresses the power management problem by scheduling a given control-dominated data flow graph. We discuss delay and power issues with scheduling, and propose an improvement algorithm for insertion of so-called soft edges which enable power optimization under timing constraints. Power savings obtained by our approach on tested circuits range between 15% and 30% of the initial power dissipation.

1: Introduction

High-level synthesis is responsible for implementing a given behavioral description onto an RTL design typically by performing the tasks of *scheduling*, *allocation* and *binding* [1]. It has been demonstrated that decisions at the behavioral level have a significant impact on power consumption of the final hardware implementation [2, 3]. In particular, for *control-dominated* designs which typically consist of lots of sequential processes, scheduling is the most critical step during the synthesis. This paper discusses how scheduling affects power consumption and presents a new technique for power optimization.

Given a *control-data flow graph* (CDFG) which is used to represent a design, scheduling is to sequence nodes (i.e., operations) in the graph by assigning each node to a specific control step (i.e., a time interval equal to the clock period). In general, the effect of scheduling on power is complex, and related to the other high-level synthesis subtasks. For instance, scheduling could be power efficient if one sequences correlated variables/operations such that they can share the same resources. A scheduling technique to enable power management was proposed in [4]. The basic idea is to schedule operations so as to identify the unused ones which can be shut down for power savings. However, the algorithm proposed in [4] suffers from several problems: (i) It ignores the active probability and computational complexity of operations, which can significantly affects power; (ii) The algorithm starts with a multiplexor which is closest to the outputs of the graph. This may not be a preferable choice, since selection of a particular multiplexor may prevent from selecting others which could have resulted in more power savings.

The goal of this work is to improve the existing technique [4] by exploring the power-manageable scheduling problem in depth. In the following section, we focus on power aspects in formulating the problem. Section 3 is devoted to the timing effect of scheduling. The algorithm and experiment are given in Sections 4 and 5, respectively. Section 6 concludes the paper.

2: Power-Manageable Scheduling Problem

We start with a CDFG which describes a design, where each node corresponds to an *operation*, and each directed edge represents *data dependency* or *control relation*. Assuming that all *conditionals* in the design are represented by multiplexor nodes, the edge entering the control input of a multiplexor node corresponds to a control relation. The node which fanouts to this edge is called *control node*, and the multiplexor is called *target* of the control node. In this section, we first show how power management is enabled during scheduling by using a simple example, and then extend our discussion to a general case where the power-manageable scheduling problem is to be formulated.

2.1: A Scheduling Example

The idea of enabling power management during scheduling was first proposed in 1996 by *J. Monteiro* [4]. To illustrate it, consider Fig. 1 which shows part of a CDFG, where *C* is a control node, *M* is a multiplexor node (i.e., the target of *C*), and nodes *a* and *b* are 0 and 1 input fanins of *M*, respectively. In general, we need to calculate *a*, *b* and *C* to obtain the output. From power optimization point of view, however, only one of *a* and *b* needs to be activated, depending on the output of *C*. This requires that node *C* is to be computed before nodes *a* and *b*. In other words, if we modify the graph by adding extra directed edges, which we call *soft edges*, from *C* to both *a* and *b* (shown as dotted lines in Figure 1) such that *C* is scheduled before *a* and *b*, then the obtained architecture would be power efficient (or *powe r manageable*) as either *a* or *b* can be shut down

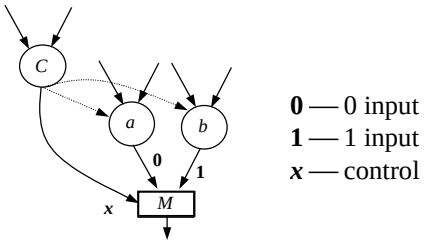


Fig. 1: Scheduling for power management

without changing the functionality of the graph. Obviously, this new architecture is subject to the latency and/or resource constraints of the design.

2.2: Problem Formulation

We define a given CDFG as $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is a set of nodes, and $\boldsymbol{E}$ a set of directed edges. $\boldsymbol{V_c} \subset \boldsymbol{V}$ is defined as a set of control nodes, i.e., $\boldsymbol{V_c} = \{C_1, C_2, \ldots, C_N\}$, where C_i is the i-th control node and N is the number of control nodes. C_i's target (multiplexor) is denoted by M_i. We use $\boldsymbol{T_i^0}$ and $\boldsymbol{T_i^1}$ to denote a set of transitive fanins of the 0 and 1 inputs, respectively, of M_i. Let $\boldsymbol{F_i} = \boldsymbol{T_i^0} \cup \boldsymbol{T_i^1}$. Generally speaking, a soft edge may be added from a control node C_i to any node $v_j \in \boldsymbol{F_i}$. We define $e(C_i, v_j)$ as follows:

$$e(C_i, v_j) = \begin{cases} 1, & \textit{if a soft edge is added from } C_i \textit{ to } v_j \\ 0, & \textit{otherwise} \end{cases} \quad (1)$$

Assume P_{C_i, v_j} is the power savings by power management due to the added soft edge (C_i, v_j), and P_{v_k} is the power savings contributed by node v_k (i.e., due to all possible soft edges which enter v_k), then P_{v_k} can be expressed as

$$P_{v_k} = \sum_i \varphi(e(C_i, v_k), P_{C_i, v_k}) \quad (2)$$

where function φ will be given later in the paper. Our objective is to find $e(C_i, v_j)$ for all $C_i \in \boldsymbol{V_c}$ and $v_j \in \boldsymbol{F_i}$ so as to maximize the total power savings:

$$S = \sum_{v_k \in \bigcup_i F_i} P_{v_k} \quad (3)$$

under the given timing and/or resource constraints which will be shown later on.

2.3: Calculation of Power Savings

In this section we look at power aspects of our problem formulation by exploring the possible power savings on different nodes due to soft edges.

The transitive fanins (denoted by $\boldsymbol{F_i}$) of a multiplexor node M_i can be potentially candidate nodes to which we can add soft edges from control node C_i. Thus, we can define a set of real candidates, $\boldsymbol{F_i'}$, for each control node C_i (where $\boldsymbol{F_i'} \subseteq \boldsymbol{F_i}$), and equation (3) can be rewritten as

$$S = \sum_{V_k \in \bigcup_i F_i'} P_{V_k} \quad (4)$$

Without loss of generality, we assume that for each multiplexor node, the probability of its output taking its 0 input is α, and the probability of its output taking its 1 input is β, where $\alpha + \beta = 1$ (these probabilities can be obtained from input statistics by high-level simulation). Also, we can associate each node v with the *weight* w_v which represents its power consumption when it is *active*. Consider the graph shown in Fig. 2. If a soft edge is added from node C to A (or B), one can reduce the power consumption by $P_{C,A} = \beta w_A$ (or $P_{C,B} = \alpha w_B$) using power management. By adding two edges from C to both A and B, power savings are given by $P_{C,A} + P_{C,B} = \beta w_A + \alpha w_B$. When one more edge is inserted from C to A_1, the total power reduction increases to $P_{C,A} + P_{C,B} + P_{C,A_1} = \beta (w_A + w_{A_1}) + \alpha w_B$, where $P_{C,A}$, $P_{C,B}$ and P_{C,A_1} are contributed by nodes A, B and A_1, respectively.

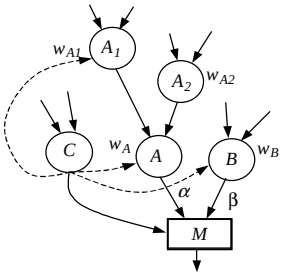


Fig. 2: Calculation of power savings due to an individual soft edge

The calculation of power savings can become complex when two or more soft edges are added from different control nodes to a same node. This is shown in Fig. 3 where up to three soft edges (i.e., dotted lines in the figure) could be added to node Q. Let us first look at what happens if only two edges (i.e., edges (C_1, Q) and (C_2, Q) in the figure) are added to Q. Individually, power savings due to edges (C_1, Q) and (C_2, Q) are $P_{C_1,Q}$ and $P_{C_2,Q}$, respectively. Since there is spatial correlation between M_1 and M_2, the total power savings are given by

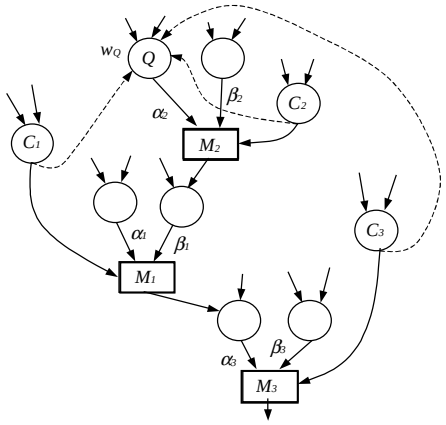


Fig. 3: Power savings due to the node (*Q*) with multiple incoming soft edges

$$P_Q = P_{C_1,Q} + P_{C_2,Q} - P_{cor(Q)} \quad (5)$$

where $P_{cor(Q)}$ is called *correlation power* which accounts for the overlapping part of power reduction by shutting down Q based on C_1 and C_2. Assuming that α_1 (or β_1) is independent of α_2 (or β_2), we have

$$P_{cor(Q)} = \alpha_1 \cdot \beta_2 \cdot w_Q \quad (6)$$

Calculation of $P_{cor(Q)}$ can be extended to the cases with more than two soft edges onto one node. For instance, if there is a third soft edge being added to node Q, as shown in Fig. 3, the power savings will be

$$P'_Q = P_{C_1,Q} + P_{C_2,Q} + P_{C_3,Q} - P'_{cor(Q)} \quad (7)$$

where

$$P'_{cor(Q)} = w_Q(\alpha_1 \cdot \beta_2 + \alpha_1 \cdot \beta_3 + \beta_2 \cdot \beta_3 - \alpha_1 \cdot \beta_2 \cdot \beta_3) \quad (8)$$

Power savings on the node with more than three incoming soft edges can be computed in a similar manner.

By defining $P_{cor(Q)} = 0$ when node Q has only one soft edge associated with it, the power savings due to any node v_k (i.e., equation (2)) can be written as

$$P_{v_k} = \sum_{i|v_k \in F'_i} e(C_i, v_k) \cdot P_{C_i,v_k} - P_{cor(v_k)} \quad (9)$$

Since adding a soft edge requires an extra control logic which comes about power penalty, we use $P_{penalty}$ to represent the total power penalty due to all these control logic parts:

$$P_{penalty} = P_{avg} \cdot \sum_{i|v_k \in F'_i} e(C_i, v_k) \quad (10)$$

where P_{avg} is the average power penalty for each soft edge. Thus, (4) can be modified as

$$S = \sum_{v_k \in \cup_i F'_i} P_{v_k} - P_{penalty} \quad (11)$$

where P_{v_k} and $P_{penalty}$ are given by (9) and (10), respectively. Eq. (11) is to be maximized subject to timing constraints that will be discussed in the following section.

3: Timing Constraints

While power management can be made by adding soft edges into a CDFG, it may degrade the timing performance of the design and/or require more hardware resources. For real applications, timing and resource constraints are specified and need to be met for the synthesis and final implementation. In an extreme case where there are no constraints, we can add as many soft edges as possible for power optimization. In this section, we discuss the impact of soft edges on timing requirements.

Given the timing (*latency*) constraints of a CDFG, each node in the graph can be assigned to a specific control step by ASAP (*As Soon As Possible*) scheduling or ALAP (*As Late As Possible*) scheduling. The ASAP and ALAP *values* of a node denote the earliest and latest control steps, respectively, which can be assigned to the node. The *slack* of any node *v* is the difference between its ALAP and ASAP values, i.e., *slack*(*v*) = *ALAP*(*v*) – *ASAP*(*v*). The CDFG meets the timing constraints and is said to be *schedulable* if and only if the slack is non-negative for all nodes, i.e.,

$$slack(v) \geq 0, \quad \forall v \in V \quad (12)$$

In general, when a soft edge is added from a control node C_i to any node $u \in \boldsymbol{F}_i'$, the slacks of u and its transitive fanouts may be reduced since their ASAP values may increase. Meanwhile, the slacks of C_i and its transitive fanins may be reduced since their ALAP values may decrease. More specifically, consider Fig. 1 again. Suppose the ASAP values of nodes *C* and *a* are *ASAP*(*C*) , *ALAP*(*C*) and *ASAP*(*a*), *ALAP*(*a*), respectively, before any soft edge is inserted. When a soft edge (say, (*C*, *a*)) is added, two cases would happen to node *a* and/or its transitive fanouts: (i) if *ASAP*(*a*) > *ASAP*(*C*), then *slack*(*a*) and hence slacks of all transitive fanouts of node *a* remain unchanged; (ii) if *ASAP*(*a*) ≤ *ASAP*(*C*), then *slack*(*a*) will be reduced by *ASAP*(*C*) + 1 – *ASAP*(*a*). This indicates that the timing constraints are met, after adding the edge, if *slack*(*a*) ≥ *ASAP*(*C*) + 1 – *ASAP*(*a*) (or, *ALAP*(*a*) ≥ *ASAP*(*C*) + 1). Whenever necessary, all slacks of node *a*'s transitive fanouts can be updated (the readers are referred to [5] for the details about slack updating).

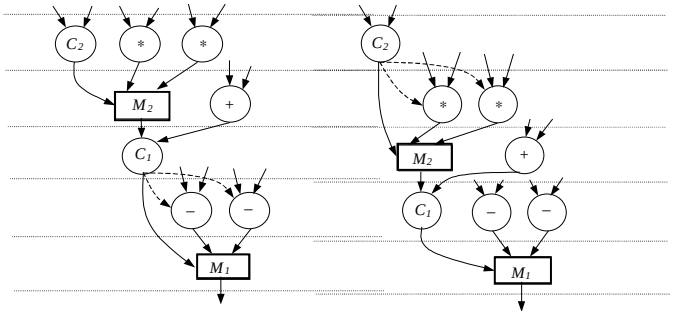
Another effect of adding a soft edge (*C*, *a*) in Fig. 1 is that the ALAP values and hence slacks of node *C* and/or its transitive fanins may decrease. Similar to the above discussion on slack, there are two cases to consider here: (i) no slack updating is required if *ALAP*(*a*) > *ALAP*(*C*); (ii) if *ALAP*(*a*) ≤ *ALAP*(*C*), then *slack*(*C*) will be reduced by *ALAP*(*C*) + 1 – *ALAP*(*a*). To guarantee the timing after adding the edge, we require *slack*(*C*) ≥ *ALAP*(*C*) + 1 – *ALAP*(*a*) (or, *ALAP*(*a*) ≥ *ASAP*(*C*) + 1). In general, how many transitive fanins of node *C* require slack updating

depends upon their topologic connectivity and ALAP values [5]. Without timing constraints, one can add a soft edge from each control node C_i to each node $v \in \boldsymbol{F_i}'$ which is a set of real candidates for C_i , as long as the power can be reduced.

4: Algorithm

Our scheduling problem is to maximize power saving (given by Eq. (11)) using power management by adding soft edges from control nodes to their candidates, subject to timing constraints (given by Eq. (12)). The existing algorithm looks at each multiplexor individually for possible insertion of soft edges, and starts with those multiplexors closer to the outputs of the graph. Intuitively, this would shut down a larger number of operations in the circuit. However, this is not always true. Fig. 4 shows such an example with two control nodes. Given timing constraints of five clock cycles in the graph, we can have two choices for scheduling. The first choice which the existing algorithm would prefer is to add soft edges from node C_1 to the two subtractors, as shown in Fig. 4 (a). Alternatively, one can add soft edges from node C_2 to the two multipliers, as shown in Fig. 4 (b). However, we are not able to add soft edges for both C_1 and C_2 since only five control steps are allowed. Obviously, scheduling in Fig. 4 (b) is a better decision in terms of power savings, for the multiplier consumes more power than the subtractor.

The above example motivates us to first look at a multiplexor with transitive fanins of high weight for effective power saving. On the other hand, whenever a soft edge is inserted from a control node to one of its target's transitive fanins, slacks of the control node's transitive fanins and its target's transitive fanouts may decrease. This is known as *slack overhead*. An ideal case is that when both ASAP and ALAP values of the transitive fanin are larger than those of the control node, respectively, there would be no slack overhead, as discussed in Section 3. Therefore, a general strategy to our scheduling problem is to maximize power savings while minimizing slack overhead under the guaranteed timing performance of the design. To this end, we propose a heuristic algorithm whose pseudo-code is described in Fig. 5. Most of this algorithm is self-explanatory. In particular, three variables (*average weight*, *slack overhead* and *priority value*) are defined in Step 2. The average weight is used to measure power savings if a soft edge is added from C_i to a candidate, the slack overhead measures the slack reduction caused by the soft edge, and the priority value denotes the combination of power saving and slack overhead. $\lambda \geq 0$ is a balance factor which is to be specified by users. The control node with maximum priority value should be investigated first for adding soft edges. After processing all control nodes, we perform physical scheduling. The scheduling result is to be used for generating final datapath and controller circuits.



(a) (b)

Fig. 4: Comparison of different ways of adding soft edges (the scheduling in (b) is better than the scheduling in (a) in terms of power savings)

Step 1: Compute ASAP and ALAP values of all nodes in a given CDFG;
Step 2: **for** each control node C_i in the graph {
while (C_i is not processed) {
Find its target node M_i ;
Identify a set of real candidates, $\boldsymbol{F}'_i$;
Define an *average weight* $AW_i = \sum_{v \in F_i'} w_v / |F_i'|$;
Define a *slack overhead*
$SO_i = ASAP(C_i) + ALAP(C_i) - ASAP(M_i) - ALAP(M_i)$;
Define a *priority value*

$$PR_i = \begin{cases} AW_i + \lambda / SO_i , & \text{if } SO_i > 0; \\ AW_i - \lambda \cdot SO_i , & \text{if } SO_i \leq 0; \end{cases}$$

}
}
Step 3: Select a control node C_k such that $PR_k = \max_i PR_i$;
Step 4: Add as many soft edges as possible from C_k to the transitive fanins of M_k subject to timing constraints;
Step 5: Label C_k as processed, and update ASAP, ALAP and slack values;
Step 6: If not all control nodes have been processed, go to Step 2;
Step 7: Execute scheduling;

Fig. 5: Pseudo-code of our algorithm

5: Experimental Results

Since no control-dominated circuit benchmarks are available to us at this stage, we used some randomly generated CDFGs for our experiments. Throughout the experiment, we also made the following assumptions: (i) There are no loops in the given CDFG, and each node has a unit delay; (ii) α and β values for all multiplexors are set to 0.5; (iii) For calculation of power savings, the weights for multiplier (∗), comparator (>), multiplexor (#), subtractor (–) and adder (+) are set to 20, 4, 1, 3 and 3, respectively; (iv) Power penalty (i.e., P_{avg} in Eq. (10)) for each soft edge

is set to 1, which accounts for the power consumption due to the extra control logic and interconnection required.

For comparison, we implemented both the existing algorithm [4] and ours (the value of λ was set to unity). Table I summarizes our experimental results for some of tested example circuits. The second column of Table I lists circuit statistics. For each circuit, we used different control steps (i.e., timing constraints) which are shown in the fourth column of Table I. Given less control steps which correspond to *tight* timing constraints (an *over-constrained* case), power savings can be as low as zero, as shown in the "0" entries in Table I. The reason is that no soft edges could be inserted without violating the timing (i.e., the number of control steps specified) requirement. On the other hand, for the cases where timing constraints are *loose* enough to allow insertion of most or all possible soft edges from control nodes to their candidates (an *over-unconstrained* case), the power savings are significant. These are shown in the last column of Table I. In general, our algorithm produces much better results than the existing algorithm except two extreme cases where the timing is either over-constrained or over-unconstrained. Power savings obtained by our algorithm on tested examples range between 15% and 30% of the initial power dissipation, compared to 2% and 24% by the existing algorithm [4]. Finally, it can be seen that our algorithm does not require more resources than the existing algorithm. This is true even for various timing constraints, as shown in the third column of Table I.

6: Concluding Remarks

We have presented our work on power-manageable scheduling by introducing soft edges into control-dominated circuits. We have discussed the impact of this technique on timing and power dissipation of high-level circuit description, and thereby proposed a heuristic algorithm for power optimization. Further work is to combine this technique with other design steps for low power synthesis.

References

[1] D. D. Gajski and L. Ramachandran, "Introduction to High-Level Synthesis," IEEE Design and Test of Computers, pp.44-54, 1994.

[2] A. Chandrakasan, M. Potkonjak, J. Rabaey, and R. Broderson, "HYPER-LP: A System for Power Minimization Using Architectural Transformations," In Proc. of ICCAD, pp.300-303, Nov. 1992.

[3] R. Mehra and J. Rabaey, "Behavioral Level Power Estimation and Exploration," IWLPD'94, pp.197-202.

[4] J. Monteiro, S. Devadas, P. Ashar, and A. Mauskar, "Scheduling Technique to Enable Power Management," In Proc. of DAC, pp.349-352, 1996.

[5] C. Chen, X. Yang, and M. Sarrafzadeh, "Potential Slack: An Effective Metric of Combinational Circuit Performance," in Proc. of ICCAD, pp. 198-201, Nov. 2000.

Circuit	Number of nodes					Resource requirement	Control steps	Number of soft edges		Power savings (%)	
	#	>	+	–	*			Alg. of [4]	Ours	Alg. of [4]	Ours
Cir. 1	2	2	1	2	2	#: 1 >: 1 +: 1 –: 2 *: 2	5	0	0	0	0
							6	2	2	1.7	30.5
Cir. 2	2	2	1	2	5	#: 1 >: 1 +: 1 –: 1 *: 2	5	0	0	0	0
							6	1	2	0.4	15.1
Cir. 3	8	8	25	3	20	#: 2 >: 2 +: 12 –: 1 *: 10	9	0	0	0	0
							10	12	12	10.9	15.6
							11	14	14	12.7	18.3
Cir. 4	4	4	5	3	8	#: 1 >: 1 +: 3 –: 2 *: 5	6	0	0	0	0
							7	3	3	5.4	18.8
							8	5	7	17.5	25.6
Cir. 5	7	7	16	14	13	#: 2 >: 2 +: 10 –: 5 *: 6	10	0	0	0	0
							11	8	10	9.6	20.6
							12	12	14	23.8	26.7

Table I: Performance comparison of existing algorithm and ours running on some CDFGs