Low Power High Level Synthesis By Increasing Data Correlation

Dongwan Shin
School of Electrical Engineering
Seoul 151-742, Korea
speanut@poppy.snu.ac.kr

Kiyoung Choi
School of Electrical Engineering
Seoul 151-742, Korea
kchoi@azalea.snu.ac.kr

ABSTRACT

With the increasing performance and density of VLSI circuits as well as the popularity of portable devices such as personal digital assistance, power consumption has emerged as an important issue in the design of electronic systems. Low power design techniques have been pursued at all design levels. However, it is more effective to attempt to reduce power dissipation at higher levels of abstraction which allow wider view. In this paper, we propose a simultaneous scheduling and binding scheme which increases the correlation between consecutive inputs to an execution unit so that the switched capacitance of the execution unit is reduced.

The proposed method is implemented and integrated into the scheduling and assignment part of the HYPER synthesis environment. Compared with the original HYPER synthesis system, average power saving of 23.0% in execution units and 14.2% in the whole circuit, is obtained for a set of benchmark examples.

1 Introduction

In the past, most work on high level synthesis has focused on techniques for area and performance optimization [1][2]. In recent years, with the increasing performance and integrity of the VLSI circuits as well as the popularity of portable devices, power consumption has emerged as an important issue in the design of electronic systems [3]. However, most of the efforts to reduce power consumption have been focused on logic, circuit, and layout levels. Relatively less research has been devoted to higher levels of abstraction which allow wider view.

Power consumption in a CMOS digital system consists of dynamic, short-circuit, and leakage power consumption. In this paper, we consider only dynamic power because short-circuit and leakage power take very small portion of the total power consumed in a system. In addition, they are rather low level issues whereas we are interested in higer levels of abstraction. Dynamic power is described by the following equation:

$$Power = \frac{1}{2} C_{sw} V_{DD}^2 f = \frac{1}{2} \alpha C V_{DD}^2 f \qquad (1)$$

where C_{sw} is the switched capacitance, V_{DD} is the supply voltage, and f is the frequency of operation. C_{sw} depends on physical capacitance C and activity factor α.

Due to the quadratic effect of voltage on power, one of the most effective ways of reducing power consumption is to reduce supply voltage, but at the cost of increased delay. Thus, a common approach to power reduction is to increase the performance of the design and then reduce the voltage as much as possible. Previously proposed methods include increasing concurrency, pipelining, and increasing the number of hardware units [4].

Other important techniques to reduce power consumption are to minimize switched capacitance. This is achieved by preserving data correlations between successive data inputs through module selection [5] and allocation [6]. In [7], Musoll and Cortadella proposed scheduling and binding that executes operations with common inputs in successive cycles on the same execution units. They obtained 7–10% power reduction in execution units. Mehra and Rabaey reduced power consumption in interconnect elements by exploiting locality of reference minimizing the accesses over global computing resources [8]. Srivastava, et al. performed a demand-driven operation and predictive powerdown to avoid wasteful transitions [9]. Abnous and Rabaey proposed a hybrid architecture exploiting application-specific units that consume less power than general-purpose ones to support programmability [10].

In this paper, we propose a simultaneous scheduling and binding scheme which increases the correlation between consecutive inputs to an execution unit so that the switched capacitance of the execution unit is reduced. The proposed

method uses DBT(Dual-Bit Type) model [11] for estimating power consumption in execution units.

This paper is organized as follows. Section 2 describes how the scheduling and binding are performed for low power. In section 3, we present the results for a set of benchmarks. Section 4 concludes the paper.

2 Scheduling and binding for low power

2.1 Motivation

The two major subtasks of high level synthesis are scheduling and binding. During scheduling, the operations in CDFG are assigned to a clock cycle(control step) where they are executed. In binding, they are assigned to specific hardware units. Scheduling and binding are tightly related to each other. In general, power consumption in datapaths accounts for a large fraction of the circuit power budget in datapath-dominated circuits. Within a datapath, execution units are major components that consume power.

The power consumption in execution units depends on switching activity of operands, that is, data correlations between successive inputs. It has been shown that more positively correlated input data stream consumes less power [4]. If we assign operations to execution units through scheduling and binding in such a way that the input data correlations increase, power consumption of execution units can be reduced.

In this paper, we propose a simultaneous scheduling and binding scheme which increases the correlation between successive inputs to the same execution units in order to reduce the switched capacitance of the execution units. The proposed method uses modified list scheduling approach for the scheduling and binding and uses DBT(Dual-Bit Type) model [11] for estimating the power consumption in execution units.

2.2 Target Architecture

In this paper, we assume the same target architecture as that used in HYPER synthesis system [12]. Each execution unit has dedicated single-ported register files at its inputs to store the variables it needs. Variables are stored into register files for the next computation when its producer operation is executed. Interconnect structure is multiplexor-based. Each Execution unit has a dedicated output buffer which can drive global buses. Multiplexors are used at the inputs of the execution units to select appropriate input buses in different time steps. The controller used for sequencing operations has a distributed structure. It consists of a global finite state machine and local decoders for the interface to datapath units.

2.3 Scheduling and binding for low power

The goal of scheduling and binding algorithm for low power is to increase the successive input data correlation in execution units. The traditional list scheduling approach has been modified to reduce power consumption in execution units. We assume that the number of resources is determined in the resource allocation step. Given CDFG $G(V,E)$, resource allocation vector **a** and latency constraint lat, the proposed method is presented as a pseudo code in Figure 1.

```
LIST_LP (G(V,E), a, lat ) {
  Compute the last possible start time t^L by
    As Late As Possible scheduling;
  Schedule operations that have the earliest t^L at control step 0;
  Put the most lately scheduled operations into L'_k;
  Determine candidate operations U'_k of which
    all predecessors have been scheduled;
  upper = 1.0;
  lower = 0.0;
  if (SOLVE(upper) is TRUE) return TRUE;
  if (SOLVE(lower) is FALSE) return FALSE;
  repeat {
    if (SOLVE((upper − lower)/2) is TRUE)
      lower = (upper − lower)/2;
    else upper = (upper − lower)/2;
  } until (upper − lower < △));
  return TRUE;
}

SOLVE(α) {
  U_k = U'_k;
  L_k = L'_k;
  repeat {
    Select type k with the lowest power cost,
      k = 1,2,...,n_res, U_k is not empty;
    Compute the switched capacitance C_k
      between l_{k,i} and u_{k,j}   ∀l_{k,i} ∈ L_k, ∀u_{k,j} ∈ U_k ;
    Select u_{k,j} operation that minimizes
      α · c_{k,ij}/C_{k,avg} + (1 − α) · t^L_j / t^L_{k,avg}  ∀c_{k,ij} ∈ C_k, ∀t^L_j ∈ t^L ;
    Schedule u_{k,j} at control step
      max{finish time of predecessors of u_{k,j},
      finish time of l_{k,i}}+1;
    Assign(Bind) u_{k,j} to the same instance as l_{k,i};
    Update the set of most lately scheduled operations L_k;
    Update the set of candidate operations U_k;
  } until (∀k, U_k is empty);
  if (lat is met) return TRUE;
  else return FALSE;
}
```

Figure 1: A simultaneous scheduling and binding algorithm for low power.

In the pseudo code, U_k is a set of candidate operations for type k. A candidate operation is an operation which is not scheduled yet, but all of whose predecessors have already been scheduled. Initially, the operations having the smallest ALAP time are scheduled at the first control step. Then, the scheduled operations are put into L_k. L_k is a set of operations each of which is an operation scheduled most lately for each instance of type k. Therefore, if we use n instances of type k, the maximum size of L_k will be n.

We process first the type that consumes lowest power. For instance, if we have types adder and multiplier, adder is selected first because it consumes less power than multiplier. By scheduling operations that consume less power first, we can have more candidate operations when we schedule operations that consume more power, thereby reducing switched capacitance more effectively. Recall that an operation becomes a candidate only when all the predecessors are scheduled.

C_k is the switched capacitance matrix computed by DBT method. Each entry in the matrix is the switched capacitance between a scheduled operation in L_k and a candidate operation in U_k. The number of entries is the number of instances of type k multiplied by the number of candidate operations of the same type. If the operation is commutative, then operand swapping is tried and the smaller switched capacitance is selected. This information is stored for register binding.

The cost of candidate operations are set to $\alpha \cdot c_{k,ij}/C_{k,\mathrm{avg}} + (1-\alpha) \cdot t_j^{\mathbf{L}}/t_{k,\mathrm{avg}}^{\mathbf{L}}$, where $C_{k,\mathrm{avg}}$ is the average switched capacitance of C_k and $t_{k,\mathrm{avg}}^{\mathbf{L}}$ is the average ALAP time of candidate operations of type k. α and $1-\alpha$ are weights of switched capacitance and ALAP time respectively. If we have tighter power constraint, we must increase α. On the other hand, if we have tighter latency constraint, we must decrease α. We perform binary search for the best value of α.

After the selection of a candidate operation $u_{k,j}$ which has the lowest cost, it is assigned the same hardware instance as the operation $l_{k,i}$ and is scheduled at the control step next to the latest among the finish steps of predecessors of $u_{k,j}$ and the finish step of $l_{k,i}$.

For a better understanding of the proposed algorithm, let's take a 7th order FIR filter as an example. Generally, FIR filter is described by the following equation:

$$\sum_{i=0}^{p-1} x_i c_i \tag{2}$$

where p is the order of the FIR filter, x_i is the input signal, and c_i is the coefficient. We assume that $p = 7$ and the values of c_0, c_1, c_2, c_3, c_4, c_5, and c_6 are -1870, 1867, -740, -1804, -740, -1867, and -1870, respectively. Figure 2 shows the Silage code and CDFG of the filter. We also assume that the latency constraint is 8 and the resource constraint is two multipliers and one adder.

The candidate operations which can be scheduled at control step 0 are the multiplication operations such as $n1$, $n2$,..., $n6$. Since $n1$ and $n2$ have the smallest ALAP time among the candidate operations, they are scheduled at control step 0 and assigned to instances $mult1$ and $mult2$, respectively(refer to Table 1). Then $n1$ and $n2$ are put into L'_{mult}. Multiplication operations $n3$, $n4$, $n5$, and $n6$ are put into U'_{mult} and addition operation $n8$ is put into U'_{add}. Since addition operation has lower switched capacitance than multiplication operation, $n8$ is scheduled first at control step 1. Then switched capacitance matrix between the most lately scheduled operations($n1$, $n2$) and candidate operations($n3$, $n4$, $n5$, $n6$) of the same type(multiplication) are computed. We can see intuitively that $n2$ and $n6$ have the highest input correlation because they have the same operand $c1$. Thus $n6$ is scheduled at control step 1 and assigned to $mult2$ instance.

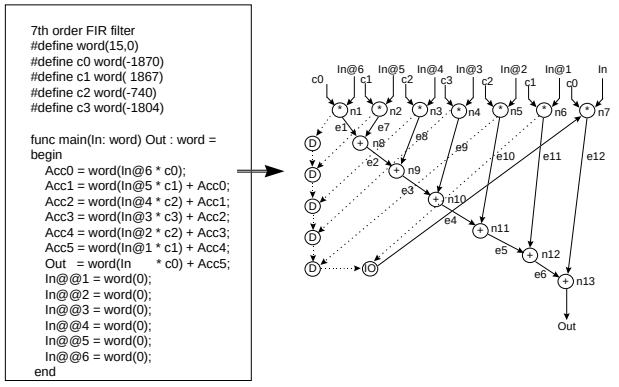

Figure 2: Silage code and CDFG of the 7th order FIR filter.

Although $n1$ and $n5$ have different operands, they have higher input correlation than any other operation and $n5$ is assigned to $mult1$ instance and scheduled at control step 1. In the same way, $n3$ which shares operand $c2$ with $n5$ is assigned to $mult1$ instance and scheduled at control step 2. Continuing this way, we obtain scheduling and binding results as shown in Table 1. The values in the parentheses are control steps where the operations are scheduled.

If the latency constraint is not met, the weight of ALAP time is increased by decreasing α. In our example, if α is decreased, $n3$ may be scheduled at control step 1 instead of $n5$. Figure 3 shows the CDFG which is scheduled and assigned by the proposed method. In this figure, gray nodes represent $mult2$ instance.

3 Experimental results

The proposed method is implemented and integrated into the scheduling and assignment part of the HYPER synthesis environment [12]. Given an algorithm and throughput constraint, the HYPER system implements an architecture that minimizes the area. The assignment uses a ran-

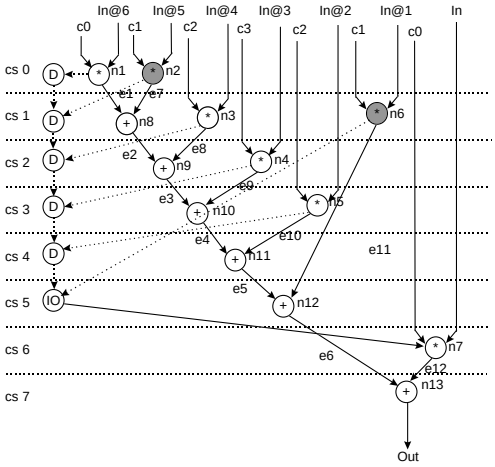

Figure 3: Scheduled and assigned CDFG of the 7th order FIR filter.

dom initial assignment with iterative improvement and the scheduling uses an enhanced list-based strategy to improve resource utilization. The allocation starts with a minimal number of units and reallocates based on the results of the assignment and scheduling. The HYPER system also provides RT level power estimation tool called SPA(Stochastic Power/Area Analysis) [11] which we used in our experiment for the evaluation of the proposed method.

3.1 7th order FIR filter

The 7th order FIR filter in Figure 2 is implemented by the HYPER system and the proposed method. The critical path in the resultant graph needs 8 clock cycles, with the assumption that both an adder and a multiplier take one cycle to execute. The latency constraint is set to 8 clock cycles. The two implementations use 2 multipliers and 1 adder. If only switched capacitance is used as the cost function, we obtain latency of 11 clock cycles. Therefore, α is decreased to meet the latency constraint.

Table 2 compares the switched capacitance, the power and the energy consumed in the two implementations. In this example, the ratio of power dissipation in execution units(EXU) to power dissipation in the whole circuit(total) is 49%. The proposed method reduces the power dissipation in execution units by 25.4%. Power dissipation in buses and registers is also decreased due to the increased data correlations. Total power reduction of 16.2% is obtained for the whole circuit.

Table 3 compares the two implementations when 2 adders are used. 9 clock cycles are obtained for the latency with the original HYPER system but 8 clock cycles are obtained with the proposed method. The HYPER system gives longer latency because it uses the list scheduling to improve resource utilization.[1]

The power reduction is 15.0% in execution units and 5.3% in the whole circuit but the energy reduction is 24.6% and 15.9%, respectively due to the reduced latency. For the comparison of implementations with different latencies, energy(or switched capacitance which is proportional to energy) is a better metric than power.

3.2 Other examples

In this section, we present results for other benchmark examples. Table 4 shows the number of operations, the length of the critical path, and the resource allocation for the benchmark examples. fir11 is an 11th order FIR filter, cascade, gm, and wf are implemetations of three different algorithms for an Avenhaus filter, dct is an implementation of discrete cosine transform, iir7 is a 7th order IIR filter, lattice is a lattice filter, and nc is a noise canceller using LMS(Least Mean Square) algorithm.

Table 5 compares the switched capacitances in execution units and in the whole circuit. The switched capacitance consumed in execution units is reduced drastically in every example(up to 50%). The switched capacitance in the whole circuit is also considerably reduced(up to 28.9%). Average switched capacitance reduction of 23.0% in execution units and 14.2% in the whole circuit, is obtained by the proposed method.

4 Conclusion

In this paper, we proposed a simultaneous scheduling and binding scheme which increases the data correlation between consecutive inputs to execution units so that the switched capacitance of the execution units is reduced. The proposed method utilizes a modified list scheduling algorithm which uses switched capacitance and ALAP time as the cost function and uses DBT(Dual-Bit Type) model [11] for estimating the power consumption in execution units.

The proposed method is implemented and integrated into the scheduling and assignment part of the HYPER synthesis environment. Compared with original HYPER synthesis system, average power saving of 23.0% in execution units and 14.2% in the whole circuit is obtained for a set of benchmark examples.

[1] Because the HYPER system is more efficient than our method in terms of resource utilization, our method may need more resources in some cases. However, we have no such case in our benchmark examples.

References

[1] D. Gajski and N. Dutt, *High-level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, 1992.

[2] G. D. Micheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw Hill, Inc., 1994.

[3] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digitial design," *IEEE J. of Solid-State Circuits*, pp. 473–484, 1992.

[4] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformation," *IEEE Tr. on CAD/ICAS*, pp. 12–31, Jan. 1995.

[5] L. Goodby, A. Orailoglue, and P. M. Chau, "Microarchitectural synthesis of performance-constrained, low-power VLSI designs," in *Proc. of Int'l Conf. on Computer Design*, pp. 323–326, Oct. 1994.

[6] A. Raghunathan and N. K. Jha, "Behavioral synthesis for low power," in *Proc. of Int'l Conf. on Computer Design*, pp. 318–322, Oct. 1994.

[7] E. Musoll and J. Cortadella, "Scheduling and resource binding for low power," in *Proc. of Int'l Symp. on System Synthesis*, pp. 104–109, Apr. 1995.

[8] R. Mehra, L. M. Guerra, and J. Rabaey, "Low power architectural synthesis and the impact of exploiting locality," *Journal of VLSI Signal Processing*, 1996.

[9] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Tr. on VLSI Systems*, pp. 42–55, Mar. 1996.

[10] A. Abnous and J. M. Rabaey, "Ultra-low-power domain-specific multimedia processors," in *Proc. of IEEE VLSI Signal Processing Workshop*, Oct. 1996.

[11] P. Landman and J. Rabaey, "Architectural power analysis: the dual bit type method," *IEEE Tr. on VLSI Systems*, pp. 173–187, June 1995.

[12] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast prototyping of datapath-intensive architectures," *IEEE Design and Test of Computers*, pp. 40–51, June 1991.

Table 1: Scheduling and binding of 7th order FIR filter.

mult1	mult2	add1	last scheduled nodes	candidates
n1(cs 0)	n2(cs 0)		n1, n2	n3, n4, n5, n6, n8
		n8(cs 1)	n1, n2, n8	n3, n4, n5, n6
	n6(cs 1)		n1, n6, n8	n3, n4, n5
n5(cs 1)			n5, n6, n8	n3, n4
n3(cs 2)			n3, n6, n8	n4, n9
		n9(cs 3)	n3, n6, n9	n4
n4(cs 3)			n4, n6, n9	n10
...	...	...	...	...

Table 2: Comparison between HYPER and proposed method of 7th order FIR filter(I)

module	HYPER			proposed			energy reduct. (%)	power reduct. (%)
	switched cap.(pF)	energy (nJ)	power (mW)	switched cap.(pF)	energy (nJ)	power (mW)		
EXU	389	9.7	12.2	290	7.3	9.1	25.4	25.4
MUX	15	0.4	0.5	20	0.5	0.6	-33.3	-33.3
BUS	137	3.4	4.3	118	2.9	3.6	13.9	13.9
CONT	46	1.2	1.4	46	1.2	1.4	0.0	0.0
CLK	24	0.6	0.8	25	0.6	0.8	-4.2	-4.2
BUFF	17	0.4	0.5	18	0.4	0.6	-5.9	-5.9
REG	80	2.0	2.5	77	1.9	2.4	3.8	3.8
total	709	17.7	22.2	594	14.9	18.6	16.2	16.2

Table 3: Comparison between HYPER and proposed method of 7th order FIR filter(II)

module	HYPER			proposed			energy reduct. (%)	power reduct. (%)
	switched cap.(pF)	energy (nJ)	power (mW)	switched cap.(pF)	energy (nJ)	power (mW)		
EXU	386	9.6	10.7	291	7.3	9.1	24.6	15.0
MUX	21	0.5	0.6	18	0.4	0.6	14.3	0.0
BUS	152	3.8	4.2	135	3.4	4.2	10.5	0.0
CONT	46	1.2	1.3	46	1.2	1.4	0.0	-7.7
CLK	36	0.9	1.0	34	0.9	1.1	5.6	-10.0
BUFF	17	0.4	0.5	18	0.5	0.6	-5.9	-20.0
REG	84	2.1	2.3	81	2.0	2.5	3.6	-8.7
total	742	18.5	20.6	624	15.6	19.5	15.9	5.3

Table 4: Characteristics of benchmark examples

benchmark	Number of operations			critical path	resource allocation
	mult(*)	add(+)	sub(-)		
fir11	11	10		11	+(2), *(2)
cascade	17	16		10	+(2), *(2)
gm	11	24	8	34	+(3), *(2), -(1)
wf	12	20	14	22	+(3), *(2), -(3)
dct	16	13	13	7	+(2), *(2),-(2)
iir7	20	10	4	11	+(2), *(2),-(1)
lattice	9	6	3	10	+(1), *(2),-(1)
nc	32	25	1	12	+(3), *(2),-(1)

Table 5: Comparison of latency and switched capacitance

benchmark	HYPER			proposed			swit. cap. reduct.	
	lat-ency	EXU (pF)	total (pF)	lat-ency	EXU (pF)	total (pF)	EXU (%)	total (%)
fir11	12	2164	4774	12	1487	3750	31.3	21.4
cascade	12	507	1279	11	427	1147	15.7	10.3
gm	34	792	3084	34	635	2819	19.8	8.9
wf	23	299	2887	23	277	2594	7.4	10.1
dct	14	170	1069	14	85	852	50.0	20.3
iir7	15	2144	5418	15	1363	3851	35.5	28.9
lattice	10	933	2305	10	798	2136	14.5	7.3
nc	24	1680	4878	23	1521	4587	9.4	6.0