DSP DATAPATH SYNTHESIS FOR LOW-POWER APPLICATIONS

Lih-Yih Chiou, Khurram Muhammand† and Kaushik Roy

School of Electrical and Computer Engineering,
Purdue University, West Lafayette, IN 47906, USA
†Texas Instruments, Dallas, TX 75243, USA

ABSTRACT

In this paper, we present a high-level synthesis technique targeting low power consumption for data- dominated applications. We have used a statistical estimation technique to obtain switching activity of modules when sharing of computing resources are required in a design. The technique enables us to understand switching behavior under resource sharing. Using the relationship between switching power and resource sharing thus obtained, we developed scheduling and allocation algorithms to reduce data path switching power. Experiments performed on various examples show up to 49% improvement in power reduction under resource constraints.

1. INTRODUCTION

Low power applications continue to challenge the VLSI industry to design power efficient circuitry. Opportunities exist to explore the design space for power and performance at different levels of design abstraction. At the architecture level, transformations and high-level synthesis have been used to address the challenging tasks of reducing power consumption [1][2][3][4][5].

Although different power reduction techniques have been proposed, most of the research in high-level synthesis only consider average power consumption as the primary power parameter to perform minimization, while in reality different functional modules have different power consumption under different input signal conditions. With the availability of input statistics in the early phases of the design, high-level synthesis can perform better trade-offs. Hence costly redesign steps can be avoided.

The primary tasks in high-level synthesis are divided into two phases: scheduling and allocation [6]. Scheduling determines at what time step an operation will be executed. Allocation consists of two subphases: resource binding and module selection. Resource binding designates an operation (or a node) to a specific hardware instance that it can be performed on. Module selection involves selecting a specific hardware implementation from a set of templates of a functional unit.

High-level synthesis typically attempts to minimize the execution time under the available resources. Recently, several research have addressed the issues of minimizing power dissipation at the behavioral level. The work in [7][5] minimized switched capacitance during allocation and resource binding, while module selection combined with voltage scaling was proposed in [2]. Scheduling with data locality was proposed in [3][4]. We use signal strength information to perform scheduling, resource binding and module selection when resource sharing is required in a design. The main idea is to reduce the signal strength difference among inputs of a shared resource.

The rest of the paper is organized as follows. Section 2 reviews signal strength based switching activity models. Section 3 describes our algorithms that take advantage of the available power characterization and signal statistics. Experimental results are reported in section 4. Finally, section 5 draws the conclusions.

2. SIGNAL-STRENGTH BASED SWITCHING ACTIVITY MODELS

Signal strength that is derived from world-level signal statistics provide us means to characterize switching activity of components with less parameters [8][9]. Models for shared and non-shared resources have been constructed and used to estimate switching activity at higher levels of design abstraction. We have found that there exists an empirical relationship between switching power and resource sharing.

2.1. Signal Strength

The signal strength, η, is defined as the number of bits needed to represent the average signal power. η is given as

$$\eta = log_2((2^{N-1} - 1) \times \frac{\sqrt{E(X^2[n])}}{d} + 1) \qquad (1)$$

where X[n] represents an input sequence to a DSP system and is assumed to be a stationary Gaussian process. E($X^2[n]$) is the average signal power of X[n] and N is number of bits used to represent the signal value. $\sqrt{E(X^2[n])}$ equals the standard deviation of zero-mean signals. All signals are assumed to be uniformly quantized in a dynamic range of $\pm d$ and are represented in sign magnitude form using N bits. With the given statistics of an input sequence (such as average, variance and correlation) we can compute η of the sequence by using Eq. 1.

An example that uses signal strength to characterize components is illustrated as follows. Consider a functional unit, FU, whose two input ports are A and B. We apply data

This research was funded in part by DARPA, by Purdue Research Foundation, and by ATT/Lucent Foundation.

sequences obtained from Gaussian distribution of different signal strengths varying from 1 to N-1 bits on two inputs of the module. The sequences to two inputs are $A(n)$ and $B(n)$, respectively. The switching activity of a functional block can be formulated as a function of η_A and η_B, i.e. signal strengths of $A(n)$ and $B(n)$.

2.2. Models for Resource Sharing

The signal strength based switching activity model has been extended for sharing of resources in [9]. The idea of sharing resources implies that a shared functional unit has to multiplex its input from different sources.

Difference in switching activity of a component (resource), which will be referred to as Δsw, is defined as the difference in switching with and without sharing. The difference, Δsw, can be positive or negative. Experimentally, we have observed an increment in switching activity in most cases for resource sharing. We observed that the Δsw of a shared functional unit is affected by the difference of η's of input sequences. Δ_η will be used as the notation for difference of η's for any two signals.

The sharing condition is based on the difference of η's at every primary input of a module. Consider the FU, that is alternately shared by two operations, OP_1 and OP_2. Each operation originally has two incoming sequences, $A_1(n)$ and $B_1(n)$ for OP_1 and $A_2(n)$ and $B_2(n)$ for OP_2. The corresponding η's are denoted as η_{a1}, η_{b1}, η_{a2} and η_{b2}. The Δ_η of two sequences into input A is computed as $D_a = |\eta_{a1} - \eta_{a2}|$. $D_b = |\eta_{b1} - \eta_{b2}|$ is for input B. Using D_a and D_b as well as η_{a1} and η_{b1}, the switching activity of the shared FU can be constructed .

2.3. Switching Activity with Resource Sharing

Next we explain the relationship between switching power and resource sharing. We first define percentage switching increment (γ) to indicate the relative switching behavior under sharing of resources. Percentage switching increment is defined as follows:

$$\gamma = \frac{\text{Difference in switching activity}(\Delta sw)}{\text{Switching Activity without sharing}} \times 100. \quad (2)$$

Figure 1 show the normalized percentage switching increment for a most significant bit(MSB)-first carry-save (CS) array multiplier [10] under resource sharing. There are five bars for every position of η_{b1}. Each represents $D_a = 1, 2, 3, 4, 9$. The sharing conditions are $D_b = 0$ while D_a is varying. The varying D_a is based on fixing η_{a1} to 6. As $D_a = |\eta_{a1} - \eta_{a2}|$, η_{a2} can take the following values: 7,8,9,10,15. We collect data from 5 different sharing conditions, $(D_a = 1, D_b = 0)$, $(D_a = 2, D_b = 0)$, ..., $(D_a = 9, D_b = 0)$. We take the data at $\eta_{a1} = 6$ and $\eta_{b1} = 1, 2, \ldots, 14$ from each sharing condition and place them side by side. We normalize bars at every position of η_{b1} by the bar whose value is the largest among the five bars. For example, bars that represent γ at $D_a = 1, 2, 3, 4, 9$ are normalized by the γ at $D_a = 9$. We observe that no matter how large η_{b1} is, there is always about 90% difference between $D_a = 1$ and $D_a = 9$. Similar behaviors can be observed for least-significant-bit(LSB)-first array multipliers,

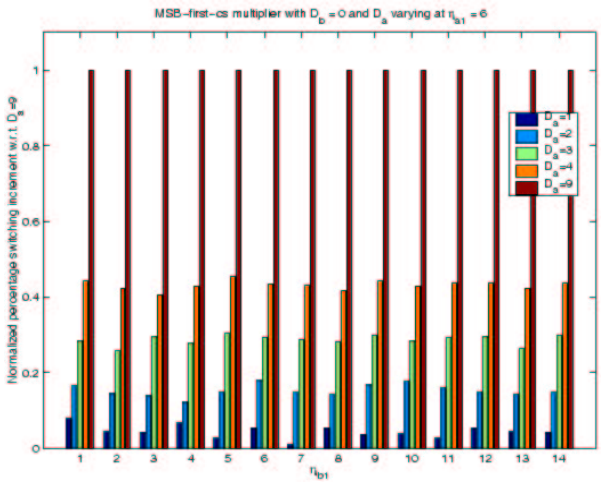


Fig. 1. Normalized Percentage switching increment (γ) for a 16-bit most-significant-bit(MSB)-first carry-save(CS) array multiplier under resources sharing when $D_b = 0$ and D_a is varying at $\eta_a = 6$

tree multipliers and adders. These results suggest that Δ_η of sequences which share the same resource has significant impact on switching activity. We will discuss how to apply these observations to high-level synthesis in following section.

3. ALGORITHMS

The observations we made from section 2 can be restated as follows: sharing a resource among two operations with higher signal similarity can result in a module that generates lower switching activity (hence, consumes lower power). With this idea in mind, we formulate the scheduling and allocation algorithm in Fig. 2. Our goal is to demonstrate that strength of input signals can be incorporated to improve designs. We focus our discussion on the sharing of two operations for one resource for demonstrative purposes. Nevertheless, we have implemented our method to perform resource sharing of n operations.

3.1. Definitions

Let us define a factor ($\mathcal{S}$) called signal similarity of two nodes (or operations) to help understand our algorithms. Signal similarity is given as

$$\mathcal{S} = \frac{1}{D_s + 1} \quad (3)$$

$$D_s = \frac{1}{k-1}\left(\sum_{i=1}^{k-1} |\eta_{a_i} - \eta_{a_{i+1}}| + \sum_{i=1}^{k-1} |\eta_{b_i} - \eta_{b_{i+1}}|\right) \quad (4)$$

where k is number of sharing for one resource. In the sequel, we will limit our discussion to $k = 2$ from now on for simplicity. All discussions for $k = 2$ can be applied to any k with some modifications. For $k = 2$, $D_s = |\eta_{a1} - \eta_{a2}| + |\eta_{b1} - \eta_{b2}|$.

The similarity, $\mathcal{S}$, indicates the closeness of two input signal profiles. Each input signal profile has two sequences that can be applied to two inputs of a node (or operation), respectively. D_s is the distance of η's (signal strength) of two signal profiles. The η's of the first signal profile are denoted as η_{a1} and η_{b1}. η_{a2} and η_{b2} are η's for two sequences of the second signal profile. The smaller the D_s is, the higher is the similarity ($\mathcal{S}$).

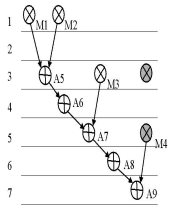
```
G ={n|nodes in DFG}
A ={a|resource assignment}
ST = {st|input signal statistics for n in G}
  1. Perform Mobility and Similarity analysis
  2. For each node n in G
       from least mobility to highest mobility, do
   2.1 If mobility(n)= 0, AssignTimeStep(n, ASAP(n)).
   2.2 Otherwise, find a time slot, p, according to
       similarity exclusive rule and AssignTimeStep(n, p).
  3. For each function type f of G, do
   3.1 Perform initial resources assignment, A
   3.2 Create a Pair List, NPL
   3.3 For each pair np in NPL
       from highest similarity to least similarity, do
   3.3.1 If none of the members in np is assigned,
       bind members of np and update A.
   3.3.2 If one of the member in np reaches
       the max number of sharing,skip to next pair.
  4. For each a in A, do
   4.1 Assign a with a module
       from library with least power consumption.
  5. Done
```

Fig. 2. Algorithm

In the following, we describe steps to perform scheduling and allocation under minimum latency constraints.

3.2. Scheduling

In step 1 mobility and similarity analysis leave us with two mobility and similarity priority lists that are needed when deciding the order of assigning nodes. In step 2 we proceed to assign time steps to nodes. The goal is to maximize the chances of operations to share modules with higher similarity to reduce overall switching activity (power consumption). Nodes with least mobility are assigned first. Nodes that have non-zero mobility are assigned to time steps subject to not only data dependence and resource constraint, but also to the similarity exclusive rule. Similarity exclusive rule keeps nodes with higher similarity from overlapping their time steps. It is better explained using the example in Fig.3. In the example, M1, M2 and M3 are already scheduled. M4 is the next one to be scheduled. Since resource constraint is 2, M4 can be assigned to any time step between 3 and 5. By similarity exclusive rule, M4 will avoid time steps 3 and 4 occupied by M3 and will be assigned to time step 5. By avoiding assigning M3 and M4 in the same time step, we can have M4 to share one functional unit with M3. Their similarity is 1, the largest in the similarity list.



(a) An example DFG

```
Resource constraint mul. = 2
Similarity:
(M1, M4) = 0.25;
(M2, M4) = 0.25;
(M3, M4) = 1;

M1 scheduled to time step 1;
M2 scheduled to time step 1;
M3 scheduled to time step 3;
M4 to be scheduled;
```

(b) conditions

Fig. 3. An example of similarity exclusive rule. Here multipliers takes two time steps to perform multiplication.

3.3. Allocation

In the resource binding phase (step 3 in Fig. 2), the goal is to maximize the number of sharing of resources while minimize the power. We use a greedy method to find pairs of operations from high similarity to low similarity. We first pair operations that are compatible and order pairs according to their similarities. Then we add one pair at a time to the solution set. The process stops when sharing is no more possible. Though this would not guarantee an optimal assignment, it will return a promising solution. In the module selection phase (steps 4 in Fig. 2), by using the signal strength switching activity tables obtained in section 2, we can scan through finite number of library modules quickly and select the one with minimal power consumption (switching activity) based on signal strength at primary inputs.

4. EXPERIMENTAL RESULTS

In this section we present the effectiveness of the proposed synthesis techniques on different benchmarks using accurate power simulator, PowerMill. We compare two design methods. First one uses our technique with the information of signal strength. The other one is implemented using conventional method without knowledge of signal statistics. Both methods have the same number of resources (i.e. functional units, registers and multiplexiers) and timing constraints.

4.1. Setup and Procedures

We obtain architectural-level designs by two different methods and synthesizes the designs from the architectural level to the circuit level. First, signal statistics are propagated from the primary inputs using the analytical method given in [11]. With the signal statistics such obtained, signal strength can be computed. Second, we apply our algorithms to perform scheduling and allocation. The prototype software has been implemented in the C++ programming language. Third, we use Synopsys Design Compiler to synthesize the VHDL code using low-power standard cell

Benchmark	Data	Conv.	Our	Reduction
FIR6	AUD1	1003.18	783.29	28.1%
	AUD2	1241.57	1033.66	20.1%
	IMG1	972.17	734.94	32.3%
FIR11	AUD1	2277.05	1661.44	37.1%
	AUD2	2891.12	2183.88	32.4%
	IMG1	2253.82	1652.99	36.3%
IIR4	AUD1	1595.65	1238.07	28.9%
	AUD2	2044.62	1750.36	16.8 %
	IMG1	1722.48	1153.38	49.3%
DIFF	SIT1	1455.81	1212.76	20.0%

Table 1. Simulated power consumption of functional units in the benchmarks. (μw)

	Data	FU	Overall
FIR6	AUD1	28.1%	17.6%
	AUD2	20.1%	15.3%
	IMG1	32.3%	18.5%
FIR11	AUD1	37.1%	23.5%
	AUD2	32.4%	22.9%
	IMG1	36.3%	23.4%
IIR4	AUD1	28.9%	11.2%
	AUD2	18.8%	7.6%
	IMG1	49.3%	26.7%
DIFF	SIT1	20.0%	14.47%

Table 2. Power reduction for different components.

library developed by Carnegie Mellon University [12]. The cell library is implemented using $0.35\mu m$ CMOS technology with 3.3V supply. In the last step we use a variety of data sequences to evaluate designs. AUD1, AUD2 and IMG1 are real audio and image data signals. SIT1 is real processing data sequences for DFF. The power consumption reported by PowerMill is the average power consumption over the entire simulation period.

4.2. Analysis of the results

Various benchmarks including FIR, IIR filters and a differential equation solver (DIFF) are used to evaluate our algorithm. All benchmark circuits have datapath width of 17 bits. Experimental results are summarized in Table 1 and 2. Table 1 reports the simulated power consumption of functional units for every benchmark. Table 2 provides power reduction using our signal-strength conscious design method. The results indicate that our design reduces power consumption of functional units up to 49% and 26% for the overall system.

It is worth mentioning that power consumption in registers can be the dominant component. Registers consume about 40-60% of the total power consumption, while functional units use up to 50%. Power consumption in control logic is less than 2%. This is partly because our current methodology does not minimize the power consumption of registers. This suggests that more efforts on minimizing the power consumption of registers are required.

5. CONCLUSIONS

We developed a behavioral synthesis method that can lower power dissipation through scheduling and allocation. Our algorithm heuristically schedules operations partly by mobility and partly by similarity of signal strength, binds resources by using a greedy approach, and select modules with least power based on signal strength. Experimental results show significant reduction in power dissipation (up to 49%) with respect to the conventional design technique.

6. REFERENCES

[1] A.P. Chandrakasan et al., "Optimizing Power Using Transformations," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 14, no. 1, pp. 12–31, January 1995.

[2] L. Goodby, A. Orailoglu, and P.M. Chau, "High-Level Synthesis Methodology for Low-Power VLSI Design," in Proceedings - IEEE Symposium on Low Power Electronics, 1994, pp. 48–49.

[3] R. Mehra, L.M. Guerra, and J.M. Rabey, "Low-power Architectural Synthesis and the Impact of Exploiting Locality," Journal of VLSI Signal Processing, vol. 13, pp. 239–258, 1996.

[4] E. Musoll and J. Cortadella, "Register-Transfer Level Transforms for Low-power Data-Paths," Integrated Computer-Aided Engineering, vol. 5, pp. 315–332, 1998.

[5] A. Raghunathan and N.K. Jha, "An ILP Formulation for Low Power Based on Minimizing Switched Capacitance During Data Path Allocation," in 1995 IEEE International Symposium on Circuits and Systems, 1995, pp. 1069–1073.

[6] D. Gajski and N.Dutt and A. Wu, and J.T. Ludwig, High-Level Synthesis, Kluwer Academic Publishers, 1992.

[7] J. Chang and P. Massoud, "Module Assignment for Low Power," in Europen Design Automation Conference(EURO-DAC), 1996, pp. 376–381.

[8] M. Lundberg, K. Muhammad, K. Roy, and S.K. Wilson, "High-level Modeling of Switching Activity with Application to Low-Power DSP System Synthesis," in IEEE International Conference on Acoustics, Speech, and Signal Processing, March 1999, pp. 1877–1880.

[9] L. Chiou, K. Muhammad, and K. Roy, "Signal Strength Based Switching Activity Modeling and Estimation for DSP Applications," VLSI DESIGN, To appear in Special Issues on Low Power Computer-Aided Design, 2001.

[10] K.Muhammand, D. Somasekhar, and K. Roy, "Switching Characteristics of Generalized Array Multiplier Architectures and their Applications to Low Power Design," in Proc. of International Conference on Computer Design: VLSI in Computers and Processors, October 1999, pp. 230–235.

[11] S. Ramprasad, N.R. Shanbhag, and I.N. Hajj, "Analytical Estimation of Transition Activity from Word-Level Signal Statistics," in Proc. of 34th ACM/IEEE Design Automation Conference (DAC), June 9-13 1997, pp. 582–587.

[12] C. Inacio, "The Carnegie Mellon Synthesizable Digital Signal Processor Core," CMU DSP Team Report, June 1999.