Data Word Length Reduction for Low-Power DSP Software

Kyungtae Han

Abstract

The increasing demand for portable computing accelerates the study of minimizing power dissipation. Most power in processes is consumed by the switching activity of capacitance. The switching power can be reduced by not only minimizing hardware but also optimizing software. Key papers for low power techniques of hardware and software are surveyed. Power analysis and minimization techniques make it possible to optimize software power. Variable word lengths in arithmetic operations without any change of hardware can be a promising software power minimization technique. The objective of this work is to analyze and minimize power dissipation of digital signal processing (DSP) blocks at software level. Preliminary results of an array multiplier with reduced word lengths and plans are presented. This research will have contribution to prolong battery life in portable DSP applications.

Index Terms

Low-power, word length, switching activity, DSP, software power minimization

I. Introduction

Portable computing demands minimizing power dissipation due to limited power supply. Since power in CMOS circuits dissipates if they are switching, a major focus of low power design is to reduce the switching activity to the minimal level required to perform the computation [1]. The switching activity is reduced by modifying hardware, changing of the operation order and reducing data word length. One of the examples of data word length reduction is shown in Fig. 1. This figure shows 4×4 bit multiplication with 2-bit multiplier and 3-bit multiplicand. This multiplication has less power consumption compared to 4-bit multiplier and 4-bit multiplicand due to unused functional blocks reduce power dissipation.

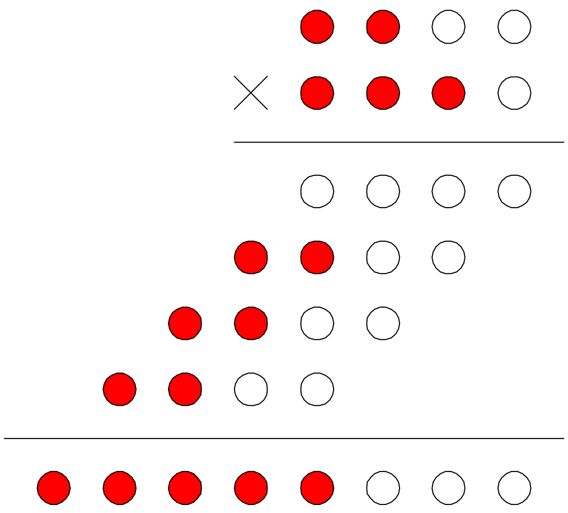

Fig. 1. Example of data word length reduction in multiply

The objective of this research is to analyze and minimize power dissipation of digital signal processing blocks at software level with reducing data word length while keeping hardware architectures. One of the problems is how to analyze and measure power consumption for software level power reduction.

After a brief summary of the power analysis in CMOS circuits, two techniques will be presented to reduce power dissipation at software level.

II. Power Analysis

There are three major sources of power dissipation in digital CMOS circuits which are summarized in the following equation: [1]

$$P_{avg} = P_{switching} + P_{short-circut} + P_{leakage} \tag{1}$$

The first term represents the switching component of power, the second term is due to the direct-path short circuit current conducting current directly from supply to ground, and the leakage power is primarily determined by fabrication technology considerations.

The switching component of power is,

$$P_{switching} = \alpha C_L V_{dd}^2 f_{clk} \tag{2}$$

where α is the switching activity parameter, C_L is the load capacitance, V_{dd} is the operating voltage and f_{clk} is the operating frequency. The switching power can be reduced through operation reduction, choice of number representation, exploitation of signal correlations, logic design, and physical design. The switching activity can be also reduced by optimizing the ordering of operations and by minimizing number of operations.

αC_L can also be viewed as the effective switching capacitance of the transistors' nodes on charging and discharging. Therefore minimizing switching activities can effectively reduce power dissipation without impacting the circuit's operational performance [2].

III. Software Power Minimization

Because of the increasing demand of software power analysis tool, Tiwari, Malik, and Wolfe [3] first systematically attempted to model software power cost. They formulated an instruction level power model for the microprocessor after measuring power of instruction sets. They made it possible to compare and evaluate programs in terms of their energy consumption for software power optimization.

Lee, Tiwari, Malik and Fujita [4] developed power analysis and minimization techniques for embedded DSP software. They found that in typical DSP applications the multiplier in the multiply and accumulate (MAC) unit is usually a major source of power consumption. A micro-architectural power model for the multiplier was developed and analyzed for further power minimization. They observed the wide current variation of `MAC` instructions mainly according to the two values being multiplied in MAC unit.

They also used the operand swapping technique for Booth multiplier [5]. The Booth multiplier does not treat two input symmetrically. Their experiment showed just swapping the operations in register A and B can reduce power for `MAC` instructions. They also used instruction packing, instruction scheduling, and memory bank assigning for low energy.

IV. Minimizing Word Length for Low-Power

Chandrakasan *et al.* [6] showed the number of bits affects all key parameters of a design, including speed, area, and power. Choi and Burleson [7] presented a general search-based methodology for wordlength optimization and used switching power model for the power dissipation. Considering a voltage dropping factor and the area of computing element according to the wordlengths, they analyzed the switching power consumption assuming the power dissipation is proportional to the area of computing element.

Erdogan and Arslan [8] showed low power multiplication schemes for finite impulse response (FIR) filter on DSP processors. They used data bus and coefficient bus for the filtering operation separately. They measured the switching activity of 8, 16, and 32 bit array multipliers for filter orders of 32, 64, and 128. They showed power reduction by decreasing in switching activity at coefficient inputs of the multiplier and both data and coefficient memory buses.

Chen, Wang and Wu [2] presented low-power 2's compelment multipliers by minimizing the switching activities of partial products using the radix-4 Booth algorithm [5]. They used the fact that switching activities of the unused functional blocks are minimized where input bits of unused functional blocks remain unaltered. They increased the probability that the partial products become zero by swapping input data.

Wordlength can be also changed by reconfigurable multiplier for low-power. Kim and Papaefthymiou [9] proposed a reconfigurable pipelined multiplier architecture by adapting its structure to computational requirements over time. It can efficiently cope with variable

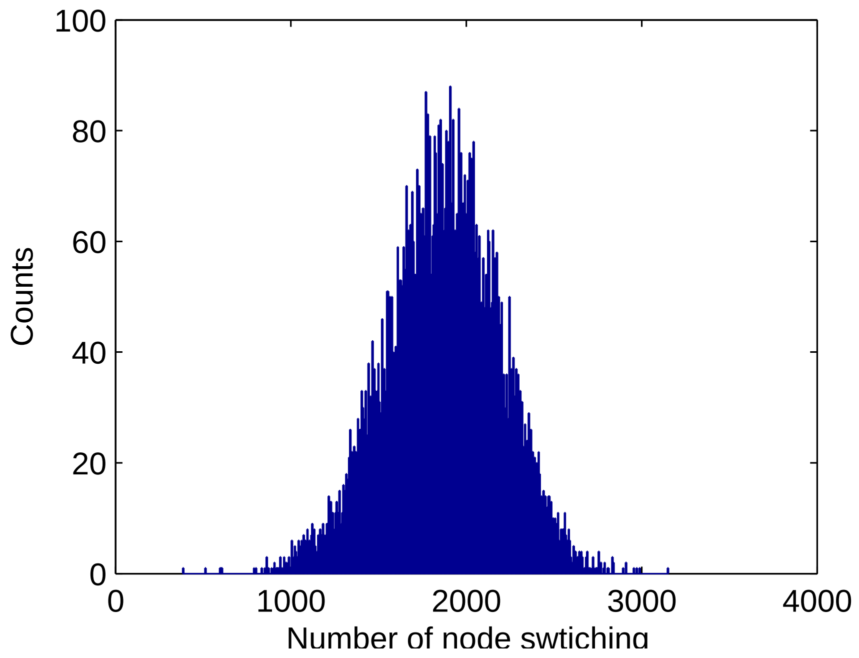

Fig. 2. Switching count of 16 × 16 bit array multiplier with 10,000 random input

data-rate multimedia applications such as video processing. The multiplier structures can dynamically reconfigure to lower their power consumption based on zero inputs and input-rate variations.

V. Preliminary Results and Plans

Much research assumed fixed hardware architectures and full dynamic range for arithmetic. However, switching activities vary according to dynamic range. For example, 16 × 16 bit multiplier has lower power consumption if input data word length is smaller. Fig. 2 shows an example of the switching result for 16 × 16 bit multiplier.

Switching activity of array multiplier is simulated using synchronous data flow (SDF) [10] since each component in the multiplier produces and consumes same number of token.

Table I shows preliminary result for reduced data word length of array multiplier with fixed hardware. The switching activity decreases as input data word length is reduced.

TABLE I
TRANSITION COUNTS FOR ARRAY MULTIPLIER OF 16 $\times$ 16 BIT MULTIPLIER WITH 10,000 RANDOM DATA

Input data	Min	Max	Std	Mean
4-bit	0	305	54	94
8-bit	0	1198	163	518
16-bit	383	3152	319	1863

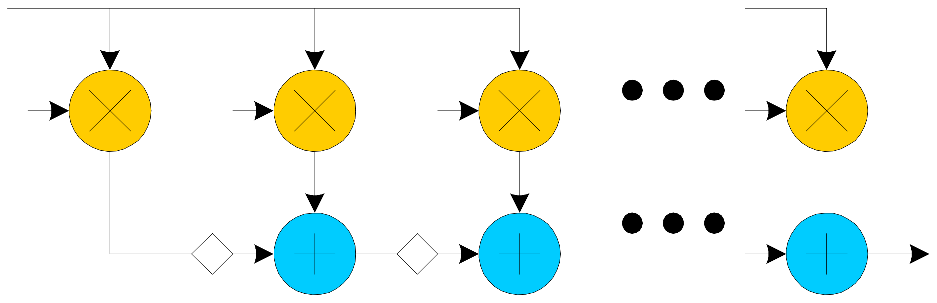

Fig. 3. FIR filter for SDF graph

Different architectures such as Booth Radix-4 multiplier and Wallace multiplier [11] will be also simulated and compared.

The power can be reduced but precision has loss, if data word length is shorter. So tradeoffs between power consumption and precision will be researched.

Power model for different input word length does not exist. So I will formulate the power model for arithmetic operations.

FIR filter will be simulated for DSP application. The filter structure is shown in Fig. 3. On each firing, every block will consume and produce a signal token. Therefore the filter will use SDF graph [10].

References

[1] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.

[2] Oscal T.-C. Chen, Sandy Wang, and Yi-Wen Wu, "Minimization of switching activities of partial products for designing low-power multipliers," *IEEE Trans. VLSI Syst.*, vol. 11, pp. 418–433, June 2003.

[3] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, San Jose, CA, Nov. 1994, pp. 429–435.

[4] M. T. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded DSP software," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 123–135, Mar. 1997.

[5] A.D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, pp. 236–240, 1951.

[6] A. P. Chandrakasan, M. Potkonjak, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 12–31, Jan. 1995.

[7] H. Choi and W. P. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proc. IEEE Workshop on VLSI Signal Processing*, Oct. 1994, vol. 7, pp. 198–207.

[8] A. T. Erdogan and T. Arslan, "Low power multiplication scheme for FIR filter implementation on single multiplier CMOS DSP processors," *Electronics Letters*, vol. 32, pp. 1959–1960, 1996.

[9] S. Kim and M. Papaefthymiou, "Reconfigurable low-energy multiplier for multimedia

system design," in *Proc. IEEE Workshop on VLSI*, Apr. 2000, pp. 129–134.

[10] B. L. Evans, *Class Notes for EE382C: Embedded Software Systems*, The University of Texas at Austin, 2003.

[11] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Comput.*, vol. 13, pp. 14–17, 1964.