Journal of the Korean Physical Society, Vol. 33, No. , January 1998, pp. S258∼S261

A Low Power Booth Multiplier Based on Operand Swapping in Instruction Level

J. H. Kim[*], J. S. Lee and J. D. Cho

Sungkyunkwan University, Suwon 440-746

We present a new low-power modified Booth algorithm with multiplier and multiplicand swapping based on a new power estimation model in instruction level. The estimation model considers a new recoding weight with the inter-instruction effects. The proposed algorithm in this paper results in a power reduction of 8 % on the average of 8 % of instances, compared with the previous result. The new low-power modified Booth multiplier has an important application to Digital Signal Processing performing many multiplication iterations.

I. INTRODUCTION

A multiplier plays an important role in various digital systems such as computer, process controller and signal processor. Designing fast and low power multipliers has long been a great theoretical and practical interest for computer scientists and engineers. Various multiplication algorithms have been proposed, and many researchers have tried to develop high speed multiplication algorithms suitable for VLSI implementation [3,4].

The shift-and-add algorithm is a familiar multiplication method. Parallel multipliers based on the algorithm, *i.e.*, array multipliers, have been widely used, and some of them are being implemented on commercial LSI chips. This type of multiplier has a regular cellular array structure of one type of basic cell and is very suitable for VLSI implementation. However, it does not operate fast enough for longer operands because its computation time is linearly proportional to the word length of operands. In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm [3,7] and the Wallace tree [4], have been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands. The modified Booth multiplier on the processor is a major source of energy consumption for Digital Signal Processing (DSP) programs. Given a pair of values to be multiplied, power should be reduced if we put the value with the lower recoding weight into the second input of the modified Booth multiplier [6,7].

In this paper, we propose a new low-power modified Booth algorithm based on operand swapping and a new estimation technique.

[*]E-mail: jhkim@nature.skku.ac.kr

This paper is organized as follows.

Section II reviews the previous modified Booth multiplier. Section III is devoted to the power analysis and estimation technique in the instruction level. In section IV, we present an operand swapping technique for low power design. Sections V and VI draw our experimental results and conclusion.

II. MODIFIED BOOTH MULTIPLIER

First, we briefly review the previous modified Booth algorithm that has been widely used [3]. It is based on the encoding the two's complements multiplier in order to reduce the number of partial products to be added [1].

This makes the multiplier faster and consumes less hardware area. The modified Booth algorithms using radix-4 is based on the partitioning of the multiplier into overlapping groups of 3-bits, as multibit recoding [2], and each group is decoded to generate the correct partial product.

The multiplier, Y, in the two's complements can be written as:

$$Y = -y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i 2^i$$

It can be rewritten using multibit algorithm as follows:

$$Y = \sum_{i=0}^{n/2-1} (Y_{2i-1} + Y_{2i} - 2Y_{2i+1})2^{2i} \quad \text{with } Y_{-1} = 0 \qquad (1)$$

The encoding of Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, $+1$, $+2$. That is, in equation (1), the terms in brackets have values in the set $\{-2, -1, 0, +1, +2\}$. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 1.

Table 1. Partial product selection and generation process.

Y_{2i+1}	$_{2i}$	Y_{2i-1}	Recoded digit	Operation on X
0	0	0	0	0X
0	0	1	+1	+1X
0	1	0	+1	+1X
0	1	1	+2	+2X
1	0	0	−2	−2X
1	0	1	−1	−1X
1	1	0	−1	−1X
1	1	1	0	0X

Recoded Digit	Operation on X
0	Add 0 to the partial product
+1	Add X to the partial product
+2	Shift left X one position and add it to the partial product
−1	Add two's complement of X to the partial product
−2	Take two's complement of X and shift left one position

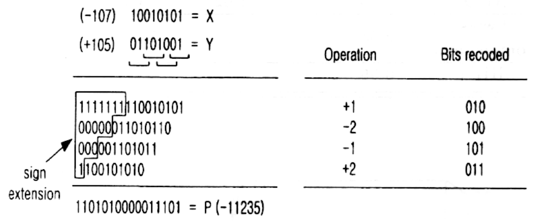

Fig. 1. 8-bit Booth multiplication example using two's complement.

The bits of the multiplier, Y, are partitioned into groups of overlapping 3-bits, each group permits the generation of a certain partial product. The five possible operations are given in Table 1.

An example is presented in Fig. 1.

The bits are grouped into 3-bit groups overlapping by one bit and a bit with a value of zero is added on the right side of Y as Y_{-1}.

So, the multiplication of two 8-bit numbers generates only 4 partial products. The partial product in this example is represented on 9-bits. For the addition of a correct partial product, the signs are extended, as shown in Fig. 1.

III. POWER ANALYSIS IN INSTRUCTION LEVEL

As a minimum power estimation, it is necessary to determine the base cost of individual instructions. Base cost refers to the portion of the power dissipation of an instruction that is independent of the prior state of the processor. Base costs for each instruction are not always adequate for a precise software power estimate. There are other energy costs that can be directly attributed to localized processor state changes resulting from the execution of a pair of instructions. These costs are referred to as circuit state or inter-operation effects. In other words, circuit state effect is the energy dissipated as a result of the processor switching from execution of one type of instruction to another. Consider the following code sequence as an illustration :

SHIFT $\quad \mathrm{B} \leftarrow \mathrm{A}$

ADD $\quad \mathrm{P} \leftarrow \mathrm{B}, \mathrm{P}$

The foremost circuit state effect among other effects on operations in X is probably the energy cost associated

Table 2. Instruction Set, Base Costs, and Circuit State Effects on 4 by 4 multiplier.

Instruction Name	E_B: Base cost [pJ]	E_S:Circuit State [pJ] when switched			
		LOAD	ADD	Two's Complement	SHIFT
LOAD	1.46	0.18	1.20	1.08	0.73
ADD	0.86		0.31	0.49	0.61
Two's Complement	0.77			0.27	0.34
SHIFT	0.29				0.15

Table 3. Instruction Set, Base Costs, and Circuit State Effects on 8 by 8 multiplier.

Instruction Name	E_B: Base cost [pJ]	E_S:Circuit State [pJ] when switched			
		LOAD	ADD	Two's Complement	SHIFT
LOAD	3.25	0.40	2.67	2.38	1.63
ADD	1.91		0.58	1.11	1.44
Two's Complement	1.72			0.55	0.78
SHIFT	0.65				0.38

Table 4. Instruction Set, Base Costs, and Circuit State Effects on 12 by 12 multiplier.

Instruction Name	E_B: Base cost [pJ]	E_S:Circuit State [pJ] when switched			
		LOAD	ADD	Two's Complement	SHIFT
LOAD	4.81	0.59	3.96	3.57	2.40
ADD	2.83		1.02	1.63	2.12
Two's Complement	2.55			1.00	1.14
SHIFT	0.96				0.78

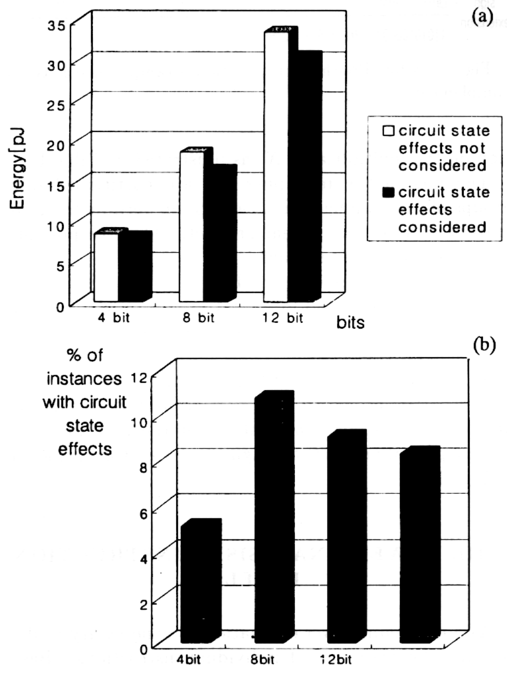

Fig. 2. Experimental Results

with the change of the instruction from a shift operation to an addition operation. The circuit state cost associated with each possible pair of consecutive instructions is characterized by measuring power supply current while executing each of the instruction pairs.

The energy dissipation in instruction level can be denoted as in the following equation:

$$\begin{aligned} E_M &= E_B + E_S \\ E_B &= \sum B_i \times N_i \quad E_S = \sum O_{i,j} \times N_{i,j} \end{aligned} \tag{2}$$

Equation (2) describes how the instruction level power measurements can be used to estimate the energy dissipation of a program. E_M is the overall energy cost of a program, decomposed into base costs (E_B), circuit state overhead (E_S), and stalls and cache misses. E_B represents base costs, where B_i is the base cost of an instruction o f type i and N_i is the number of type i instructions in the execution profile of a program. E_S represents circuit state effects where $O_{i,j}$ is the cost incurred when an instruction of type i is followed by an instruction of type j. $N_{i,j}$ is the number of occurrences where instruction type i is immediately followed by instruction type j.

Tables 2∼4 show the experimental results on the base energy and circuit state effect energy for each instruction pair. In this measure, we assumed that the multiplier's 4-bit input is {1001}. Its recoded digit is {+1,-1}. The total energy is shown in Table 5.

The base cost and the circuit state effects for each instruction are taken from Table 2. In Table 5, the two's complement instruction was preceded by LOAD instruction. Thus, the circuit state effects for the two's complement can be found at the intersection of the column labeled "two's complement" and the row labeled "LOAD". Similarly, the circuit state effects are found at the intersection of instruction to each other. Finally, the base and circuit state costs are summed to obtain the energy cost of the code sequence.

IV. INSTRUCTION ORDERING AND SWAPPING FOR LOW POWER

In this section, we present the operand swapping for low power associated with the base cost and circuit state effects. Operand swapping means to swap two operands to minimize the switching activity associated with the operation. For example, the number of additions, shifts and complements performed in the modified Booth multiplier depends on the bit pattern of the second operand (*i.e*, multiplier) [5]. The number of additions, shifts and complements required is defined as the recoding weight of the second operand. If an operand is known to have a higher recoding weight, then the operand is placed in the first operand position. If an operand is known to have a lower recoding weight, then the operand is placed in the second position. However [6], revealed that the recoding weight estimation is not accurate and has 5 % errors [6]. does not consider the circuit state effects. Therefore, we present a new model of the recoding weight, considering the circuit state effects of instructions. Then, the total recoding weight of two input operands is as follows:

$$W = \sum_i W_i$$

Table 5. The example of total energy estimation in 4 by 4 multiplier.

Recoded Digit	Instruction Set	Base Cost [p.J]	Circuit State Effects	
+1	LOAD	1.46		
	ADD	0.86	1.20	
-1	LOAD	1.46	1.20	
	Two's complement	0.77	1.08	
	ADD	0.86	0.49	
	Total	5.41	3.97	Total energy=5.41+3.97=9.38[p.J]

$$W_i \;=\; W_b + W_s,$$

Where W_i is the weight of individual recoded digit i in Booth multiplier. Here, W_b is the base cost of instructions in the recoded digit and W_s is the circuit state cost of instructions in the recoded digit. If an operand has lower W, the operand is placed in the multiplier. For example, if the recoded digit is $\{+2\}$, then the total recoding weight W_i is:

$$W_i = W_s + W_a + W_{s\to a}$$

W_s is the base cost of shift instruction. W_a is the base cost of addition instruction. $W_{s\to a}$ is the inter-operation cost between shift and addition instructions.

V. EXPERIMENTAL RESULTS

Our experiments used Synopsis$^{\text{TM}}$ and HSpice$^{\text{TM}}$. We generated a netlist using Synopsis$^{\text{TM}}$, and estimated the power consumption using HSpice$^{\text{TM}}$. We also used the carry-save-adder. Tables 2∼4 are the results of 4, 8 and 12-bitinputs, respectively. Also we made an experiment on all cases for multiplication's input.

Fig. 2(a) shows the energy level of each input instance. The white bar corresponds to the case without considering the circuit state effects. Whereas, the gray bar corresponds to the case considering the circuit state effects. There was 8 % reduction in energy consumption on average. For test set, we generated 2^n pattern input instances for n-bit (n=4, 8, 12).

Fig. 2(b) is the percentage of instances where the energy reduction is caused by the circuit state effects. As shown in Fig. 2(b), we achieved power reduction of 8 % of instances on average considering the circuit state effects.

VI. CONCLUSIONS

In this paper we proposed a new low-power modified Booth algorithm based on operand swapping with a new power estimation in instruction level. The proposed modified Booth multiplier uses a new recoding weight measure considering the circuit state effects. The new low-power modified Booth multiplier can be effectively used for Digital Signal Processing (DSP) applications. That is, in DSP applications, we can incorporate the register allocation scheme into our algorithm to further minimize the switching activity in shared registers.

REFERENCES

[1] A. Chandrakasan and R. Brodersen, *Low-Power CMOS Design* (IEEE Press, 1998), p. 218.

[2] H. Sam and A. Gupta, IEEE Trans. on Computers **39**, August (1990).

[3] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design Circuits and Systems* (Kluwer Academic Pub., 1995).

[4] R. I. Hartley and K. K. Parhi, *Digit-Serial Computation* (Kluwer Academic Pub., 1995).

[5] W. Nebel and J. Mermet, *Low Power Design in Deep Submicron Electronics* (Kluwer Academic Pub., 1997).

[6] M. T. C. Lee, V. Tiwari, S. Malik and M. Fujita, Fujitsu Scientific and Technical Journal **31**, 215 (1995).

[7] C. J. Nicol and P. Larsson, *Proceedings 1997 International Symposium on Low Power Electronics and Design*, 76 August (1997).