Design and Implementation of the Tree-like Multiplier

Gi-Yong Song, Jae-Jin Lee, Ho-Jun Lee, and Ho-Jeong Song

Dept. of Computer Engineering, Chungbuk National University

Cheongju, Chungbuk 361-763 Korea.

Tel: +82-43-261-2452, Fax: +82-43-262-2449

E-mail: gysong@cbucc.chungbuk.ac.kr

Abstract *This paper proposes a 16-bit × 16-bit multiplier for 2 twos-complement binary numbers with tree-like structure and implements it on a FPGA. The space and time complexity analysis shows that the 16-bit Tree-like multiplier represents lower circuit complexity and computes more quickly than both Booth array multiplier and Modified array multiplier.*

1. Introduction

Multiplication is one of the basic operations in various digital systems. An array multiplier[1], one of the well-known combinational multipliers, has a good repeatability of cells and is favorable for VLSI implementation. In a multiplier based on redundant binary arithmetic [2-4], additions are performed in a constant time but excessive gates are required to process the redundant binary representation. In this work we propose a 16-bit multiplier with tree-like structure, implement it on a FPGA, and then analyze it in terms of circuit complexity and computation time.

2. Array Multipliers

Booth array multiplier is a typical combinational multiplier performing multiplication on twos-complement binary numbers.

The Booth multiplier shows high degree of repeatability of cell and carries out operations concurrently to reduce the total computation time. The cell arrangement of 4-bit Booth multiplier with simplified data flow is shown in Fig.1.

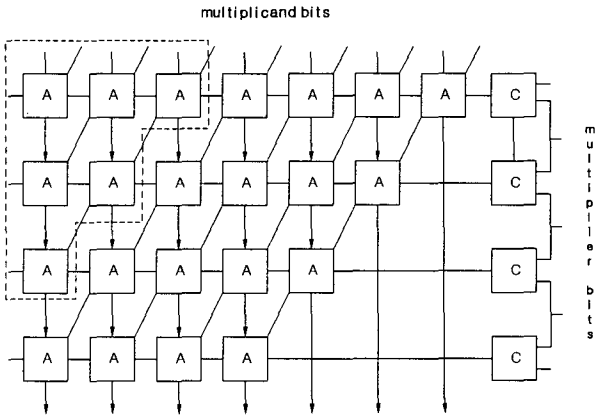


Fig. 1. A 4-bit Booth array multiplier

The cell C is for Booth recoding and cell A is for addition and subtraction. The cells in area enclosed by dotted line are introduced to cover for sign-extension required by computation on negative operand, and the number of rows for n-bit multiplier is n.

The reduction in the number of additions and subtractions from Booth recoding does not contribute to this array because there should be a row of cells for each bit of Booth-recoded multiplier representing potential addition or subtraction. To reduce the height of array, n, the canonical signed digit recoding or radix-4 recoding should be considered. Let us examine the canonical signed digit representation which is obtained from identifying isolated 1s and 0s in the multiplier first. Only a single addition or subtraction needs to be performed at an isolated 1s or 0s. The rule for forming canonical signed digit is shown in Table 1.

multiplier bit-pair		mode-in	Booth bit	mode-out
0	0	0	0	0
0	1	0	1	0
1	0	0	0	0
1	1	0	$\bar{1}$	1
0	0	1	1	0
0	1	1	0	1
1	0	1	$\bar{1}$	1
1	1	1	0	1

Table 1. The rule for forming canonical signed digit

The mode introduced to identify the isolated 1s and 0s in multiplier by detecting run of 1s and 0s is exactly the same as the carry-out of the full adder when the adjacent bit pair of multiplier and mode-in are viewed as inputs to full adder, so we can recode the multiplier into canonical signed digit representation parallelly in one step instead of going through the multiplier one bit at a time using carry-lookahead circuit.

Although the canonical signed digit representaion contains an average of *2n/3* 0s, each bit pair in the canonical signed digit form represents potential addition or subtraction and this results in the *n/2* rows of cells in the modified array. On the other hand, the radix-4 recoding examines three adjacent bits of multiplier and then generate a single recoded value located at corresponding position according to the Table 2 shown below.

multiplier adjacent bits			bit-recoding at corresponding position	
0	0	0		0
0	0	1		1
0	1	0		1
0	1	1	1	
1	0	0	$\bar{1}$	
1	0	1		$\bar{1}$
1	1	0		$\bar{1}$
1	1	1		0

Table 2. The rule for radix-4 recoding

The radix-4 recoding contains an average of *5n/8* 0s, but the number of potential addition or subtraction is *n/2*, one operation for each bit pair of multiplier, so the situation in the radix-4 recoding is the same as in the canonical signed digit recoding except for the difference due to the recoder circuit.

The recoder for canonical signed digit representation, shown in Fig.2a determines mode-out using carry-lookahead circuit and then generate the code from the converter which takes inputs from the multiplier bits and carry look-ahead circuit.

The recoder for radix-4 representation shown in Fig.2b generate the code from the three adjacent bits of multiplier. The C_1, C_2 in Fig. 2 are converter cells for each case. For *n*-bit multiplier, the recoder for canonical signed digit form in Fig.2a requires *n* converter cells in contrast to the recoder for radix-4 code requiring *n/2* converter cells.

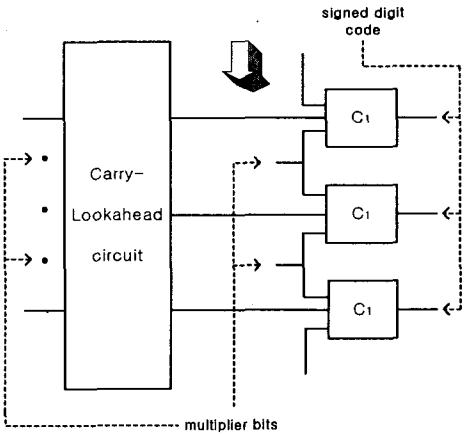


Fig. 2a. The recoder for canonical signed digit form

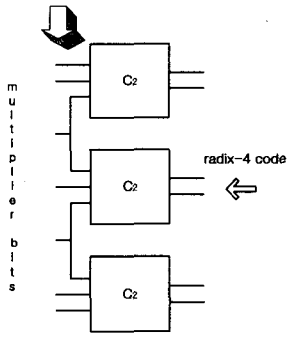


Fig. 2b. The recoder for radix-4 recoding

The complexity of the radix-4 recoder is lower than that of the canonical signed digit recoder, so the reasonable recoding selection for an multiplier will be the radix-4 recoding.

The cell arrangement of 8-bit modified array multiplier based on radix-4 recoding with simplified data flow is shown in Fig.3.

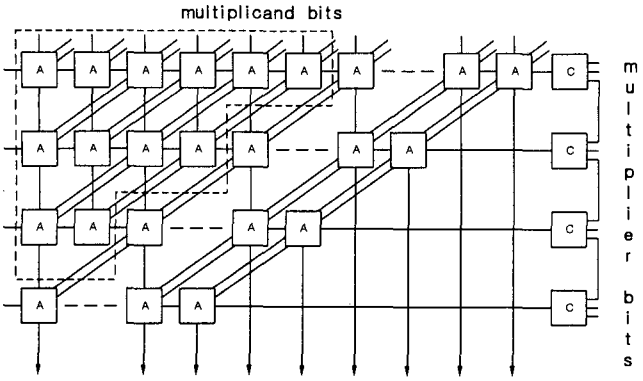


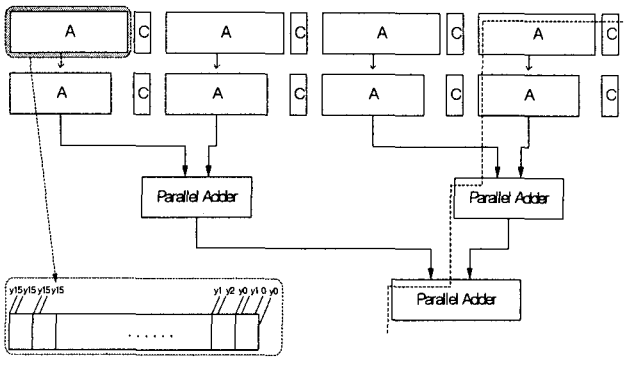
Fig. 3. Modified array multiplier

The number of rows of the array is reduced to *n/2*, but the cells for sign-extension are still required to accommodate the computations on the negative operands.

3. Design and Analysis of the Tree-like Multiplier

The cells for sign-extension will be eliminated and the height, the number of rows, will be reduced again by separating each pair of rows, and distributing each pair along the tree-like structure. The layout of 16-bit multiplier for 2 twos-complement binary number with tree-like structure is shown in Fig.4.

Fig. 4. A 16-bit Tree-like multiplie



r

Two top rows produce partial products for each pair of recorded multiplier bits and add them concurrently, then the sum of partial products feeds into the parallel adder below.

The accumulated sum of partial products is generated from each of parallel adder in the middle and then are added again by entering the lower parallel adder.
The dashed line shows propagation path which determines the multiplication time.

Fig.5 shows the implementation of 16-bit × 16-bit tree-like multiplier synthesized on a FPGA, Spartan xcs40-pq240 [5].

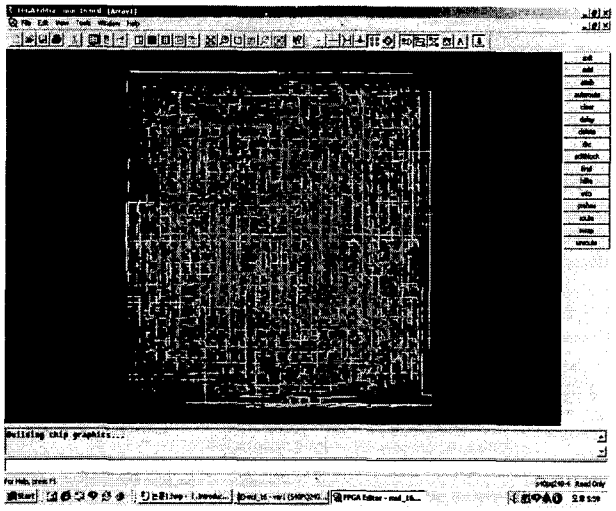


Fig. 5. Implementation of the Tree-like multiplier

We will analyze the complexity of each multiplier, and find out propagation delay which becomes computation time. The size of operand, *n*, is assumed to be a power of 2. Each combinational multiplier consists of two or three kinds of cells. These include cell capable of adding and subtracting, one-bit full adder cell, and cell for recoding. In addition that the recoder cells are simple combinational logic, the number of cells used for recoding in each multiplier is relatively small compared to the number of arithmetic cells and the effect of those cells on the circuit complexity and propagation delay is not significant, so we will regard the recoder cell as sharing the same complexity with full-adder cell and only count the number of cells without classifying them according to the logic function they implement. Most of the cells comprising each combinational multiplier are one of the two types of arithmetic cells; cells for addition and subtraction, or cells for addition only.

Let the cell for addition and subtraction have unit circuit complexity and assume the same unit

propagation delay for all cells.

Let k be the ratio of the complexity of adder cell to complexity of adder /subtractor cell. Then k becomes a constant between 0 and 1.

The complexity of each n-bit multiplier is shown in Table 3.

	Number of cells	Propagation
Booth array multiplier	$\frac{3}{2}n^2 - \frac{n}{2} + kn$	$3n - 1$
Modified array multiplier	$\frac{3}{4}n^2 + \frac{1}{2}kn$	$\frac{5}{2}n - 1$
Tree-like multiplier	$\frac{1}{2}n^2 + n + k[\frac{1}{4}n^2 + \frac{1}{2}n\log n - \frac{7}{4}n + 1]$	$2n$

Table 3. The complexity of each multiplier

4. Conclusions

In this paper the 16-bit × 16-bit twos-complement binary number multiplier with tree-like structure based on radix-4 recoding was proposed and implemented on a FPGA. The original Booth array multiplier, Modified array multiplier and Tree-like multiplier were analyzed from a standpoint of circuit complexity and computation time. In comparison with the original and modified multiplier, the Tree-like multiplier represents lower circuit complexity, although the exact complexity depends on the specific value of k, and computes more quickly.

References

[1] Hayes, J.P., *Computer Architecture and Organization*, 3rd Ed. McGraw Hill.

[2] K.W.Shin, B.S.Song and K.Bacrania, "A 200-MHz Complex Number Multiplier Using Redundant Binary Arithmetic." *IEEE J. Solid-State Circuits*, vol.33, pp.904-909, June 1998.

[3] S.M.Yen, C.S.Laih, C.H.Chen and J.Y.Lee, "An Effident Redundant-Binary Number to Binary Number Converter." *IEEE J. Solid-State Circuits*, vol.27, pp 109-112, Jan.1992.

[4] Y.Harata, Y.Nakamura, H.Nagase, M.Takiga- wa and N.Takagi, "A High-Speed Multiplier Using a Redundant Binary Adder Tree." *IEEE J. Solid-State circuits*, vol.SC-22, pp28-33, Feb.1987.

[5] Xilinx, 1999 *Xilinx Data Book*.