최적화된 4진/8진 혼합 MAC 설계

정 진 우, 김 승 철, 이 용 주, 이 용 석
연세대학교 전기전자공학과
전화 : 02-2123-2872 / 핸드폰 : 011-9768-5628

An Optimized Hybrid Radix MAC Design

Jinu Jung, S.C. Kim, Y.J. Lee, Y.S. Lee
Dept. of Electrical & Electronic Eng., Yonsei University
E-mail : jjinu@dubiki.yonsei.ac.kr

Abstract

This paper is about a high-speed MAC (multiplier and accumulator) design applying radix-4 and radix-8 Booth's algorithm at the same time.

The optimized hybrid radix design for high speed MAC has taken advantage of both a radix-4 and a radix-8 architectures. A radix-4 architecture meets high-speed, but it takes much more power and chip area than a radix-8 architecture. A radix-8 architecture needs less power and chip area than the other, but it has a bottleneck of generating three times the multiplicand problem. An optimized hybrid architecture performs the radix-4 multiplication partially in parallel with the generation of three times the multiplicand for use of the radix-8 multiplication. It reduces the concerned bit width of multiplier in radix-8 multiplication.

I. Introduction

Current computing environments have developed tremendously and now incorporate high level interactive software, user-friendly interfaces, voice/image processing and multimedia processing, all of which require high performance data processing units. It is very difficult for a general purpose processor to perform such a high degree of multimedia calculations.

This paper presents an optimized MAC, which is the most important component of the DSP. Since it determines the performance of the total DSP, it is important to optimize the MAC. We adapt the hybrid radix-4/radix-8 Booth's algorithm to improve the MAC.

II. Booth's Algorithm

A MAC involves two basic operations : the generation of partial products and their accumulation. Thus, there are two ways to speed up MAC operation. First, by reducing the number of partial products, and second, by accelerating their accumulation.

The reduction of partial products reduces the complexity of their accumulation, which thereby enables the partial products to accumulate faster.

2.1 Modified Booth's algorithm

Using a modified Booth's algorithm, the multiplier is encoded in units of 2 or 3 bits. Encoding the multiplier using a fixed number of bits regulates the time of calculation. Thus, it is appropriate for a synchronized circuit.

The Encoding of the multiplier in units of 2 or 3 bits is called the radix-4 or radix-8 modified Booth's algorithm, respectively.

The following figure 1 shows the multiplier scanning in a radix-4 modified Booth's algorithm.

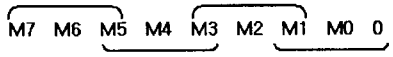

Figure 1 Multiplier scanning in a radix-4 modified Booth's algorithm.

Tables 1 and 2 show the radix-4 and radix-8 modified Booth's encoding, respectively.

III. Accumulation of the Partial Products

So far, we have been looking at the Booth's algorithm for high speed multiplication. Now, we will look at fast addition algorithms for the partial products produced by Booth's algorithm.

To improve multiplication speed, either the number of partial product must be reduced, or the time elapsed for accumulation must be reduced.

M_{i+1}	M_i	M_{i-1}	Encoded Multiplier
0	0	0	0
0	0	1	+X
0	1	0	+X
0	1	1	+2X
1	0	0	-2X
1	0	1	-X
1	1	0	-X
1	1	1	0

Table 1 Radix-4 modified Booth's encoding

M_{i+2}	M_{i+1}	M_i	M_{i-1}	Encoded Multiplier
0	0	0	0	0
0	0	0	1	+X
0	0	1	0	+X
0	0	1	1	+2X
0	1	0	0	+2X
0	1	0	1	+3X
0	1	1	0	+3X
0	1	1	1	+4X
1	0	0	0	-4X
1	0	0	1	-3X
1	0	1	0	-3X
1	0	1	1	-2X
1	1	0	0	-2X
1	1	0	1	-1X
1	1	1	0	-1X
1	1	1	1	0

Table 2 Radix-8 modified Booth's encoding

To reduce the number of partial products, Booth's algorithm is employed. The Wallace/Dadda algorithm is widely used in fast multiplication to shorten the partial product accumulation time.

3.1 Accumulation of the partial products

After generation of the partial products, all of which are accumulated to obtain the final results. If high speed accumulation of partial products is desired, a fast multi-operand adder is needed.

Figure 2 shows a conventional partial product accumulation method.

A					0	1	0	1	1	0	22
X					0	0	1	0	1	1	11
Y					0	1	0	-1	0	-1	Encoded multiplier
1	1	1	1	1	1	0	1	0	1	0	
0	0	0	0	0	0	0	0	0	0		
1	1	1	1	0	1	0	1	0			
0	0	0	0	0	0	0	0				
0	0	1	0	1	1	0					
0	0	0	0	0	0						
0	0	0	1	1	1	1	0	0	1	0	

Figure 2 Conventional partial products accumulation method

If two's complement numbers are obtained by generating the one's complement and then adding a carry to the LSB(least significant bit), there will be some hardware overhead. However, it may be minimized by realizing that the two's complement

number

$$s \quad s \quad s \quad x_4 \; x_3 \; x_2 \; x_1 \; x_0$$

is equal to

$$-s \cdot 2^7 + s \cdot 2^6 + s \cdot 2^5 + x_4 \cdot 2^4 + x_3 \cdot 2^3 + x_2 \cdot 2^2 + x_1 \cdot 2^1 + x_0$$

Since

$$-s \cdot 2^7 + s \cdot 2^6 + s \cdot 2^5$$
$$= -s \cdot 2^7 + s \cdot (2^6 + 2^5)$$
$$= -s \cdot 2^7 + s \cdot (2^7 - 2^5)$$
$$= -s \cdot 2^5$$

It may be replaced by

$$0 \; 0 \; (-s) \; x_4 \; x_3 \; x_2 \; x_1 \; x_0$$

To represent the value '-s' in the order of 4, the original sign digit 's' is complemented, and 1 is added. This way, we obtain the '-s' as required, and add 1 to column 5. The latter will serve as the extra 1 needed in column 5 to deal with the sign bit of the second partial product. Thus, an extra 1 is needed only for the first partial product.

A smaller matrix of bits to be added will result if the original sign bits are complemented and the extra 1 is added to column 5. Figure 4 shows *reduced partial product accumulation.*

					1					
					0	0	1	0	1	0
				1	0	0	0	0	0	
			0	0	1	0	1	0		
		1	0	0	0	0	0			
	1	1	0	1	1	0				
1	0	0	0	0	0					
0	0	0	1	1	1	1	0	0	1	0

Figure 3 reduced partial product accumulation

The equation below is about the manipulation of the above algorithm.

$$s_0(2^{11} - 2^5) + s_1(2^{11} - 2^6) + s_2(2^{11} - 2^7) + s_3(2^{11} - 2^8)$$
$$+ s_4(2^{11} - 2^9) + s_5(2^{11} - 2^{10})$$
$$= (s_0 + s_1 + s_2 + s_3 + s_4 + s_5)2^{11} - s_0 2^5 - s_1 2^6 - s_2 2^7 - s_3 2^8 - s_4 2^9$$
$$- s_5 2^{10}$$
$$= (s_0 + s_1 + s_2 + s_3 + s_4 + s_5 - 1)2^{11} + \sum_{i=5}^{10} 2^i + 2^5 - s_0 2^5 - s_1 2^6$$
$$- s_2 2^7 - s_3 2^8 - s_4 2^9 - s_5 2^{10}$$
$$= (s_0 + s_1 + s_2 + s_3 + s_4 + s_5 - 1)2^{11} + 2^5 + (1 - s_0)2^5 + (1 - s_1)2^6$$
$$+ (1 - s_2)2^7 + (1 - s_3)2^8 + (1 - s_4)2^9 + (1 - s_5)2^{10}$$
$$= 2^5 + (1 - s_0)2^5 + (1 - s_1)2^6 + (1 - s_2)2^7 + (1 - s_3)2^8 + (1 - s_4)2^9$$
$$+ (1 - s_5)2^{10}$$

If the second partial product is shifted two positions relative to the first partial product, the carry generated by adding 1 to $(1- s_1)$ for the first partial product is not positioned in the same column as $(1- s_2)$, the complement of the sign bit of the second partial product. Therefore, an extra 1 in column 6 is needed. Together with the carry generated by column 5, it produces the necessary 1 in column 7.

The equation below explains the existence of extra '1' in the hybrid radix architecture.

$$s_0(2^{11} - 2^5) + s_1(2^{11} - 2^7) + s_2(2^{11} - 2^{10})$$
$$= (s_0 + s_1 + s_2)2^{11} - s_0 2^5 - s_1 2^7 - s_2 2^{10}$$
$$= (s_0 + s_1 + s_2 - 1)2^{11} + \sum_{i=5}^{10} 2^i + 2^5 - s_0 2^5 - s_1 2^7 - s_2 2^{10}$$
$$= (s_0 + s_1 + s_2 - 1)2^{11} + 2^5 + (1 - s_0)2^5 - s_1 2^6 + (1 - s_2)2^7 + 2^8$$
$$+ 2^9 + (1 - s_5)2^{10}$$
$$= 2^5 + (1 - s_0)2^5 - s_1 2^6 + (1 - s_2)2^7 + 2^8 + 2^9 + (1 - s_5)2^{10}$$

Ⅳ. Proposed Architecture

The conventional radix-4 and our proposed hybrid radix architectures are shown as below. Figure 4 shows the structure of conventional radix-4 MAC and delay time. Figure 5 shows our proposed hybrid radix MAC and delay time.

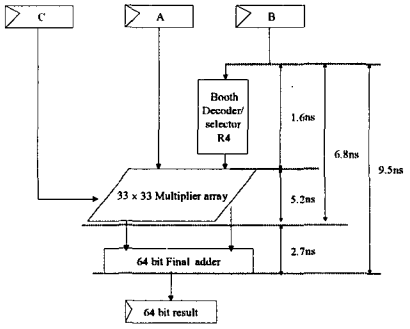

Figure 4 Block diagram of conventional radix-4 MAC

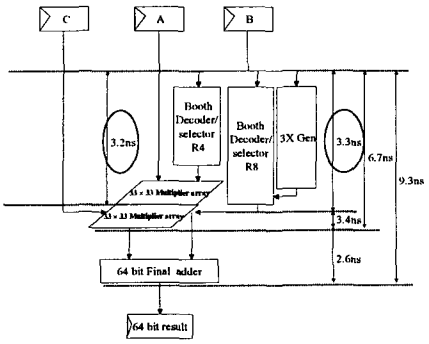

Figure 5 Block diagram of hybrid radix MAC

A hybrid radix architecture performs radix-4 multiplication partially in parallel with generation of three times the multiplicand for use of radix-8 multiplication. It reduces the concerned bit width of multiplier in radix-8 multiplication.

The Wallace tree of the conventional radix-4 shown in figure 6 has 19 partial product rows, and that of our proposed hybrid radix architectures shown in figure 7 has 15 partial product rows. So we reduced the number of partial products.

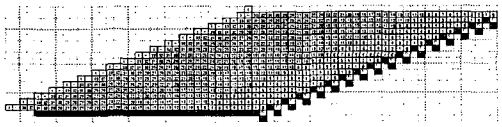

Figure 6 Wallace tree of radix-4 MAC

Figure 7 Wallace tree of hybrid radix MAC

The Wallace tree of a conventional radix-4 architecture consists of the maximum of 19 operands. Thus, using CSA (carry save adder) tree, they can be added in 6 stages of CSA tree. On the other hand, the Wallace tree of a hybrid radix architecture consists of the maximum of 4 and 13 operands at each stage. Thus, they can be added in 7 stages, 2 and 5 stages in each of CSA tree.

At the result, hybrid architecture has one more CSA tree stage than radix-4 architecture. Because of the size of radix-4 architecture, however, it must have buffers to drive the whole circuit. After synthesis, the radix-4 architecture has more area and path delay.

V. Conclusions

An optimized hybrid radix MAC unit in this paper is modeled by Verilog HDL. The verified HDL models are synthesized with a 0.35μm standard cell library using Synopsys tool under the worst case condition of 3.0V, worst case process, and 85℃ temperature.

The area is evolved by reported in the number of equivalent gates. The hybrid radix MAC is about 12,000 equivalent gates and operates over 100 MHz clock speed. Therefore, the hybrid radix MAC is optimized 32-bit MAC architecture than conventional radix-4 MAC.

Reference

[1] Brian S. Cherkauer, Eby G. Friedman, "A Hybrid Radix-4/Radix-8 Low Power, High Speed Multiplier Architecture for Wide Bit Widths", IEEE, 1996

[2] Israel Koren, "Computer Arithmetic Algorithm", A K Peters Ltd, 1993