A NEW VLSI ARCHITECTURE OF PARALLEL MULTIPLIER BASED ON RADIX-4 MODIFIED BOOTH ALGORITHM USING VHDL

RASHMI RANJAN

Department of Electrical and Electronics Engineering

Noida Institute of Engineering & Technology

Greater Noida, 201301

Email: jha.rashmi11@gmail.com

PRAMODINI MOHANTY

Department of Electrical and Electronics Engineering

Noida Institute of Engineering & Technology

Greater Noida, 201301

Email: pramodini_swain@gmail.com

Abstract:

Low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance systems. Optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. In our project we try to determine the best solution to this problem by comparing a few multipliers.

This project presents an efficient implementation of high speed multiplier using the shift and adds method, Radix-4 modified Booth multiplier algorithm. The parallel multipliers like radix 2 and radix 4 modified booth multiplier does the computations using lesser adders and lesser iterative steps. As a result of which they occupy lesser space as compared to the serial multiplier. This is very important criteria because in the fabrication of chips and high performance system requires components which are as small as possible

Keywords: Multiplier and accumulator, booth algorithm, Digital Signal Processing (DSP) and standard design.

INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters [1], microprocessors, digital signal processors, etc. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system [2]. Furthermore, it is generally the most area consuming [3]. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, a whole spectrum of multipliers with different area-speed constraints has been designed with fully parallel. Multipliers at one end of the spectrum and fully serial multipliers at the other end. In between are digit serial multipliers where single digits consisting of several bits are operated on. These multipliers have moderate performance in both speed and area. However, existing digit serial multipliers

have been plagued by complicated switching systems and/or irregularities in design. Radix- 2^n [4] multipliers which operate on digits in a parallel fashion instead of bits bring the pipelining to the digit level and avoid most of the above problems. They were introduced by M. K. Ibrahim in 1993[5].

Ai	Ai-1	Y	Comments	Explanation
0	0	0	Middle of 0's	String of 0's shift only
0	1	1.B	End of 1's	Add and Shift
1	0	-1.B	Beginning of 1's	Add and Shift
1	1	0	Middle of 1's	String of 1's shift only

Table 1: Booth's recoding algorithm Radix-2

These structures are iterative and modular. The pipelining done at the digit level brings the benefit of constant operation speed irrespective of the size of' the multiplier. The clock speed *is* only determined by the digit size which is already fixed before the design is implemented.

COMPLEMENT REPRESENTATION

In complement representation, numbers are represented as two's complement in the binary section. In this method, positive number is represented in the same way as signed-magnitude method. It is most widely used method of representation. Positive numbers are simply represented as a binary number with '0' as sign bit. To get negative number convert all 0's to 1's , all 1's to 0's and then add '1' to it. Suppose, a number which are in 2's complement form and we have to find its value in binary, then if number starts with '0' then it is a positive number and if number starts with '1' then it is a negative number.

If, number is negative take the 2's complement of that number, we will get number in ordinary binary. Let us take, 1101. Take the 2's complement then we will get 0011. As, number is started with '1' it is negative number and 0011 is binary representation of positive 3. So, the number is -3. Similarly, we are representing other negative numbers in 2's complement representation.

Suppose we are adding +5 and -5 in decimal we get '0'. Now, represent these numbers in 2's complement form, then we get +5 as 0101 and -5 as 1011. On adding these two numbers we get 10000. Discard carry, then the number is represented as '0'

In this signed multiplication we had modified the Complex Multiplication strategy, normally we are having Four Multipliers and three adder/subtractor blocks. But, in modified strategy we require Three Multipliers and five Adders.

BOOTH'S RECODING ALGORITHM

Parallel Multiplication using basic Booth's Recoding algorithm technique based on the fact that partial product can be generated for group of consecutive 0's and 1's which is called as Booth's recoding. These Booth's Recoding algorithm is used to generate efficient partial product. These Partial Products always have large number of bits than the input number of bits. This width of partial product is usually depends upon the radix scheme used for recoding. These generated partial products are added by compressors as explained in section 3.2. So, these scheme uses less partial products which comprises low power and area.

There are two types of algorithm Radix-2 and Radix-4 to generate efficient partial products for multiplication. First we will explain basic technique of Booth's Recoding algorithm and then Modified Booth's Recoding technique for Radix-2 algorithm.

BASIC TECHNIQUE OF BOOTH'S RECODING ALGORITHM FOR RADIX-2

Booth has proposed Radix algorithm for high speed multiplication which reduces partial products for multiplication. The Booth's algorithm for multiplication is based on this observation. To do a multiplication A*B, where

A= an ,an-1…..a0 is a multiplier
B= bn ,bn-1…..b0 is a multiplicand
then, we check every two consecutive bits in A at a time:-

RADIX-2

Suppose A is Multiplier having value -5 and B is Multiplicand having value +2 then, B=> 0010 (+2)
A=> 1011 (-5)

After looking into above table for multiplicand, first we see two LSB values and then adjacent values in A. We get partial product as:-

i) For 10 we have to perform -1.B, i.e., 2's complement of B, 1110.
ii) For 11 we have to put all 0's i.e., 0000.
iii) For 01 we have to perform 1.B, i.e., value of B,0010
iv) For 10 again -1.B, i.e. 1110.
Here, some bits are encapsulated called as correction bits to match the width of partial products.

```
1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0
1 1 1 1 0
-----------------
1 1 1 1 0 1 1 0      <= P
```

MULTIPLIER UNIT

As explained in previous chapters about various technique of Complex Multipliers, we found that implementation of Complex Multipliers are implemented using more than one number of Basic Multipliers are required, i.e. to implement normal way to implement Complex Multiplication, four Basic Multipliers are required. To make Complex Multiplier as low power unit, this Basic Multipliers are designed by using Compressor technique. If, the Basic Multiplier is designed as low power then Complex Multiplier also becomes a low power unit.

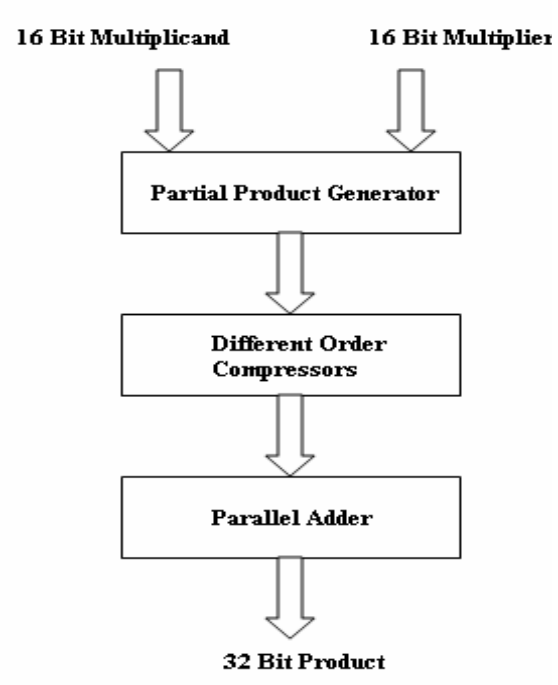

Figure 1: Internal Block Diagram of 16*16 Basic Multiplier [2]

The above figure shows Internal Block Diagram of Basic Multiplier. It consists of three stages:-

i) Partial Product Generator
ii) Different Order Compressors
iii) Parallel Adder

Below is the description of all three blocks that are used for multiplication. **Partial Product Generator:-**

In Unsigned Multiplier, normally we are generating partial products and adding them to generate result of multiplier. Let 'A' and 'B' are two n-bit unsigned numbers which is generating product 'Z' which is of 2n-bit. First we are generating Partial products by using 'AND' operation. For n bit number multiplication n*n number of partial product generated.

Let us take two 16-bit numbers A15-A0 called Multiplicand and B15-B0 called Multiplier as inputs of multiplier, partial products are generated by ANDing each bit of 'A' with each bit of 'B', so 16*16=256 number of partial products are generated. Each bit of multiplicand is ANDed with every bit of multiplicand. a0 is ANDed with b0-b15 producing m00-m015 sixteen partial product for first row. Similarly, for other 14 rows we are using AND operation of a1-a15 with b0-b15 for producing other 240 remaining partial products i.e. from m01-m1515.

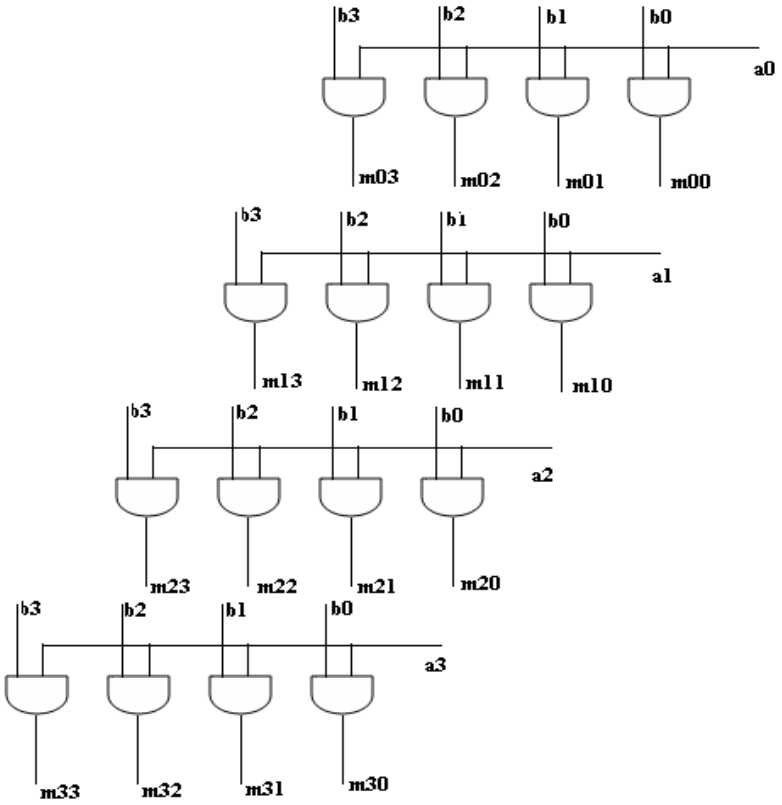

Figure 2: Partial Product Generator(4 Bit)

In above diagram Partial Product Generator is explained. a0 bit which is multiplicand is ANDed with other bits of multiplier b0-b3 producing sixteen partial products m00-m33. This Partial Products is going to the inputs of Compressors to compress the partial product stages. This Compressors are used to reduce the stages of partial products into only two stages.

SIGNED MULTIPLICATION

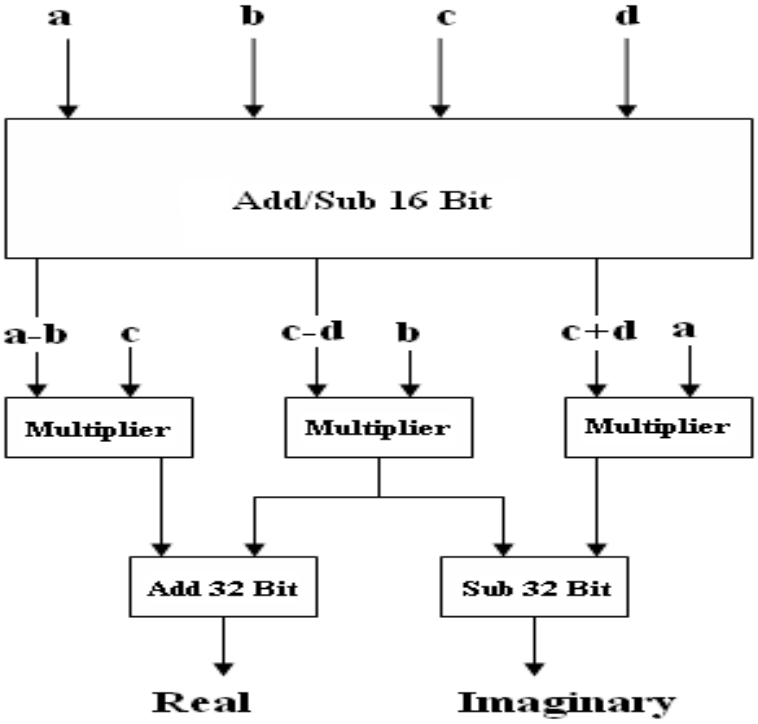

Figure 3: Modified Complex Multiplier Block Diagram.

Above Block Diagram shows Modified Complex Multiplier which consists of three multipliers and three adder/subtractor unit. These multiplier requires one less multiplier compare to previous technique. So, it consumes less power. To perform signed multiplication we are using Booth's Radix algorithm. Booth's Radix algorithm reduces partial products as compared to normal multiplier algorithm. So, it reduces the switching operation of the multiplier, hence reduces power. It is based on the fact that partial product can be generated for group of consecutive zeros & ones which is called as Booth's recoding.

MODIFIED TECHNIQUE RECODING ALGORITHM FOR RADIX-2 [5][6]

Parallel Multiplication using basic Booth Recoding Technique is explained in previous section. Since this technique requires lot of adders as a result it requires more power & area. In next proposed multiplier design, we have reduced number of adders required in partial product addition. Hence, reduction of vertical length of Partial Products. In these technique, mainly correction bits are reduced This is done without compromising correctness of multiplication of 2's complement numbers. We have used Multiplexer based Booth Recoding scheme to reduce the length and width of partial products.

In these technique, change in scheme results in partial products which after recoding are always greater than input bit length by one bit Radix-2 scheme. Similarly, in Radix-4 scheme recoding are always greater than input bit length by two bits. These additional bit/bits are act as a correction bit/bits to get correct value of the multiplier. Also, at hardware realization of Booth's recoding scheme, we can remove extra select line, which is used at the time of recoding. Because of this extra select lines multiplexer size become large. We have observed that if we do not consider this extra bit at the time of hardware realization we can reduces size of one multiplexer. So, in radix 2 LSB decides first partial product. Also, in radix 4 first two LSB bits decides first partial product. Now these partial products have been added using proposed array of adders to achieve correct multiplication output. The working of this novel design has been explained in following sections.

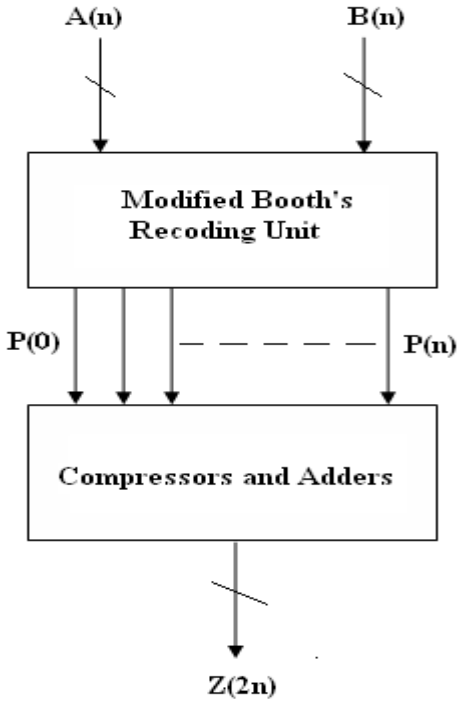

Figure 4: Block Diagram of Modified Booth's Recoding unit Multiplier [5]

In order to achieve signed number multiplication Partial Products are generated using Modified Booth's Recoding Unit Multiplication block. After generation of new Partial products these are added using Compressors and Parallel adder. Below is the explanation of Modified Booth's Recoding Unit for Multiplier.

MODIFIED BOOTH'S RECODING UNIT[5]

Partial Products are generated using Modified Booth's Recoding Unit block. As, we saw in previous section generation of Partial Products for basic Booth's Recoding algorithm, using the same concept we are generating partial products for Modified Booth's Recoding Algorithm having the length of partial product more than input bit sequence by one for Radix-2 scheme and by two for Radix-4 scheme.
These modified technique is explained below:-

RADIX-2 METHOD

As, we saw in Table 1. output partial products are added and shifted according to input sequence. Here, we are using multiplexers to generate recoding unit. Select lines of multiplexers are input bits of multiplier and outputs are according to modified table as shown below:-

Ai	Ai-1	Y	Explanation
0	0	0	All 0's
0	1	1.B	[B(n-1) , B]
1	0	-1.B	[B(n-1) , (-B)]
1	1	0	All 0's

Table 2: Modified Booth's Recoding Algorithm

RADIX 2

This can be explained with simple example:-

Suppose B => 1100 (-4)

A => 1010 (-6)

So, according to table as shown above we will obtain recoding bits as partial products:-

PP0 => 0 0 0 0 0
PP1 => 0 0 1 0 0
PP2 => 1 1 1 0 0
PP3 => 0 0 1 0 0

Here, in Modified Booth's Recoding algorithm one extra bit is added to the MSB of the input bit sequence as shown in Table. The hardware realization for this recoding unit is based on multiplexers and include 2's complement unit. At the time of recoding we are assuming one extra bit '0' before the LSB of input bit sequence and these extra bit '0' decides Partial Product according the sequence as explained in Table given above. We have observed that at the time of hardware realization only LSB is sufficient to get partial products, because of these multiplexer become 2x1 rather than 4x1 and other multiplexers will remain same as per their input select lines depending upon recoding scheme. So, multiplexers are important hardware for Booth's Recoding unit.

RESULTS

- ✓ Studied the operation of array multiplication using the Booth algorithm understands about signed number multiplication wrote hierarchical VHDL code for the multiplier.
- ✓ Synthesize the multiplier using the Xilinx ISE tools and simulated using ModelSim
- ✓ Study the synthesis performance parameters (delay, power consumption, equivalent gate count).

In addition, we compared the proposed architecture of radix4 booth algorithm with that of radix2. Because of the difficulties in comparing other factors only delay is compared in case of radix 4 and radix 2. The delay of ours was 2.68ns while it was 3.94 ns, which means that ours improved about 32% of the speed performance. This improvement is mainly due to the final adder. The architecture should include a final adder with the size of 2 to perform an multiplication. It means that the operational bottle neck is induced in the final adder no matter how much delay.

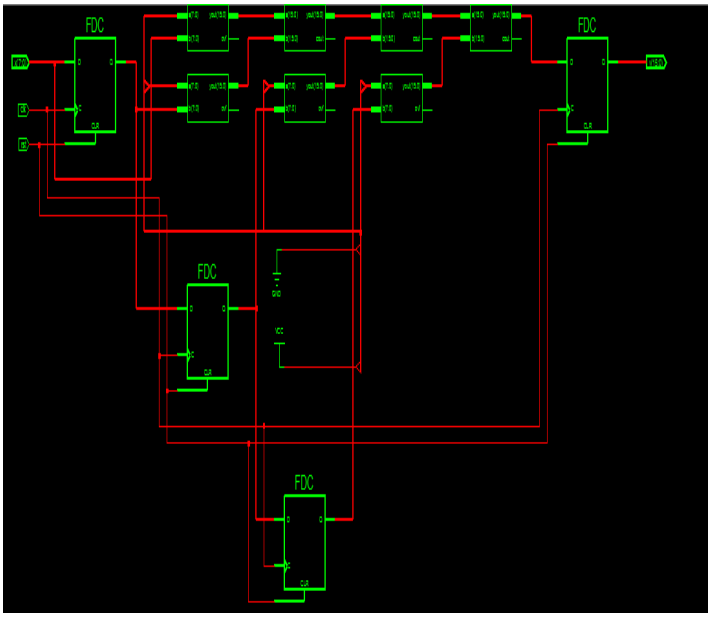

Figure 5: RTL Sematic of Booth Algorithm Radix4

In radix 4 booth algorithm architecture the peak memory usage is 139MB. The number of slice flip-flop is 33. FFs are used which increases speed. Total Fan Out used in this architecture is 55.

CONCLUSIONS

This work presents an efficient implementation of high speed multiplier using the shift and adds method, Radix-4 modified Booth multiplier algorithm. The parallel multipliers like radix 2 and radix 4 modified booth multiplier does the computations using the lesser adders and lesser iterative steps. As a result of which they may occupy lesser space as compared to the serial multiplier. This is very important criteria because in the fabrication of chips and high performance system requires components which are as small as possible.

REFERENCE

[1] John G.Proakis, Dimitris G. Monolakis, "Digital Signal Processing,Principles,Algorithms, and Applications", Fourth Edition.
[2] A. Dandapat, S. Ghosal, P. Sarkar, D. Mukhopadhyay (2009), "A 1.2-ns16×16-Bit Binary Multiplier Using. High Speed Compressors", International Journal of Electrical, Computer, and Systems Engineering, 2009, 234-239.
[3] J. Gu, C.H.Chang (2003), "Ultra low voltage low power 4-2 compressor for high speed multiplications". Circuits and Systems, 2003.ISCAS '03. Proceedings of the International Symposium, vol. 5, May 2003, 321-324.
[4] Israel Koren, *Computer arithmatics algorithms* A.K.Peters Ltd. ISBN 1568811608.
[5] A.D.Booth, A signed binary multiplication technique, Quarterly Journal of Mechanics and Applied mathematics, vol-IV,pt-2-1951.
[6] C.H.Chang, J.Gu, M.Zhang (2004) ,"Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits", Circuits and Systems Regular Papers, IEEE Transactions page(s): 1985- 1997, Volume: 51, Issue: 10, Oct. 2004.