International Journal of Engineering Research and General Science Volume 3, Issue 1, January-February, 2015
ISSN 2091-2730

Implementation of splitter based multiplier for unsigned and signed number

Subahanal Nuzrath[1] , Dhanabal[2], Sabari[3]
[1] Assistant Professor, [2,3] PG Scholar
Department of Electronics and Communication Engineering
Jay Shriram Group of Institutions
Tirupur-638660.

Abstract — Multiplier is one of the key arithmetic operations used in most of the high performance digital systems including FIR filters, digital signal processors and microprocessors. With advances in technology, many researchers have tried and are trying to design multipliers which offer either high speed or low power consumption. The proposed multiplier is finding out the best trade off solution between high speed and low power consumption. The splitter based multiplier produces n/2 partial products and overcome many disadvantages of look-up table multiplier, Wallace tree multiplier and booth multiplier.

Keywords — key arithmetic, FIR filter, digital signal processor, microprocessor, high speed, low power, partial products

INTRODUCTION

Design of efficient Multipliers is an active research topic because recent applications demand technologies to employ Digital Signal Processors (DSPs) rather than Microprocessors. In the DSPs the basic building block is the Multiply and Accumulate (MAC) unit. The MAC unit performs two processes namely multiply the multiplicant with multiplier and Accumulate (Add) the partial products to produce the desired result. Though by doing these two processes there may be time complexity to produce the result. So still there is research going on regarding MAC unit to reduce the time complexity. While discussing about the reduction of time the Booth Multiplier will reduce the partial products by assigning weights to the Multiplier and uses overlapping Method which will produce the result faster than the traditional Binary Multiplier. Even though Booth Multiplier reduces the time complexity, use of overlapping method in Booth Multiplier will be more time consuming because of the use of 2's complement. The speed of multiplication operation is of great importance in digital signal processing as well as in the general purpose processors today.

In the past multiplication was generally implemented via a sequence of addition and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them. The basic multiplication principle is twofold i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix

EXISTING MULTIPLIERS OVERVIEW

look-up table multiplier

Look-Up Table multipliers are simply a block of memory containing a complete multiplication table of all possible input combinations. The large table size will needed for even higher inputs, which make these impractical for FPGAs. For example 2 X 2 multiplication sixteen memory elements are needed to store all possible value of products whereas for 3 X 3 multiplication totally sixty four memory elements are needed to store all possible value of product result.

Wallace tree multiplier

The Wallace tree multiplier is considerably faster than a simple array multiplier because its height is logarithmic in word size, not linear. However, in addition to the large number of adders required, the Wallace tree's wiring is much less regular and more complicated. As a result, Wallace trees are often avoided by designers, while design complexity is a major concern to them. Wallace tree styles are generally avoided for low power applications, since excess of wiring is likely to consume extra power. While subsequently faster than Carry-save structure for large bit multipliers, the Wallace tree multiplier has the disadvantage of being very

irregular, which complicates the task of coming with an efficient layout. The summing of the partial product bits in parallel using a tree of carry-save adders became generally known as the Wallace Tree.

Booth multiplier

There are two algorithms used in booth multiplication They are radix 2 and radix 4 algorithm. To perform radix-2 booth algorithm following steps to be followed

Step 1: Making the Booth table

- From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.
- Let X = multiplier & Y = multiplicand
- Load the X value in the table.
- Load 0 for X_{-1} value it should be the previous first least significant bit of X
- Load 0 in U and V rows which will have the product of X and Y at the end of operation.

Step 2: Booth Algorithm

Booth algorithm requires examination of the multiplier bits, and shifting of the partial product. Prior to the shifting, the multiplicand may be added to partial product, subtracted from the partial product, or left unchanged according to the following rules

Table 1 partial product rules for Radix-2

X	X_{-1}	Operation performed
0	0	Shift only
1	1	Shift only
0	1	Add Y to U, and shift
1	0	Subtract Y from U, and shift

Similarly to perform radix-4 booth algorithm the following steps should be followed

- **Radix number conversion** → First of all the multiplier had to be converted in to radix number
- **Partial product array formation** → The multiplicand is multiplied with five different weights and are stored in an array
- **Partial product selection** → Based on three bit combination of multiplier the partial product is selected from the array.
- **Wallace tree addition** → After selecting the partial products added them using Wallace tree adder to produce product result

Drawbacks Of Existing Multipliers

By analyzing the functionality of three basic multipliers namely Look-Up Table Multiplier, Wallace Tree Multiplier and Booth Multipliers the following disadvantages are founded, they are as follows

- Memory Requirements
- Circuit Complexity
- Radix Number Conversion
- Overlapping Concept
- Number Of Weights Used
- Signed Number Multiplication

PROPOSED ARCHITECTURE

The block representation of the splitter based multiplier comprises of three blocks namely

- Multiplier splitter
- Multiplier Splitter Based Partial Product Generator (MSBPPG)
- Wallace Tree Adder
- Sign detector

In which MSBPPG is the main building block of Splitter Based Multiplier. This block also comprises of two sub blocks such as partial product array and 4-1 Multiplexer. This proposed multiplier is designed to reduced number of partial products that is n/2 partial products are generated for n inputs and also it computes the product comparatively faster than the existing multiplier.

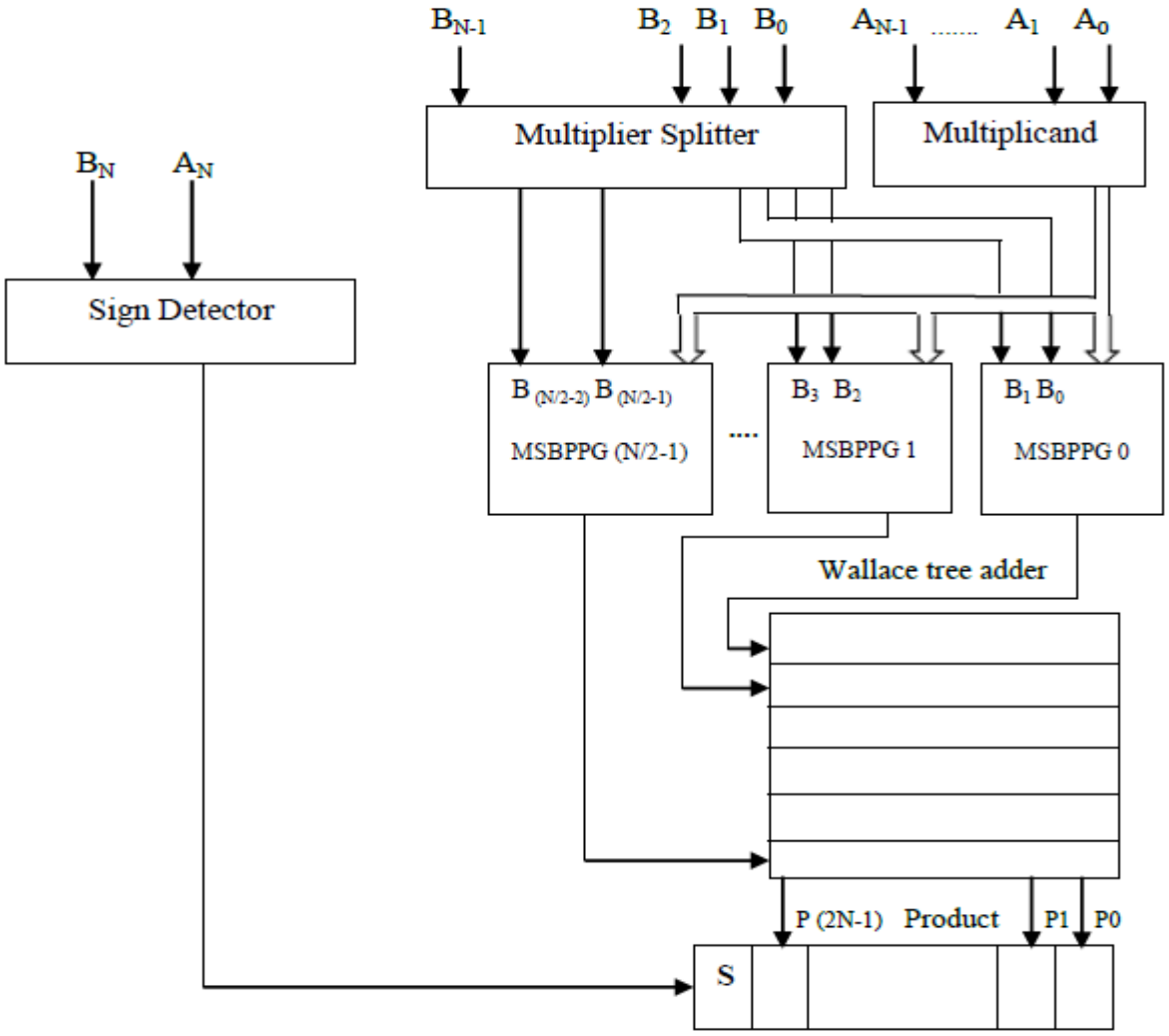

Figure 1 splitter based multiplier

Multiplier splitter

The function of multiplier splitter block is to splitting given multiplier in to two bits respectively. For multiplying 8 X 8 binary numbers the Multiplier (B) has been divided into 4 Partitions such that 4 Partial Products will be generated using this Multiplier instead of 8 Partial Products in conventional multipliers.

Multiplier Splitter Based Partial Product Generator (MSBPPG)

Multiplier splitter based partial product generator (MSBPPG) is the basic building block of splitter based multiplier, this block comprises of two sub-blocks namely, partial product array and 4-1 multiplexer. The multiplicant (A) is directly given as the

input of partial product array. The partial product array is designed to produce the number of feasible partial products by multiplying the multiplicant (A) with 4 different weights such as 0,1,2,3 based on the multiplier splitter.

In this Partial Product Array multiply by '0' means is multiplied by '0'.Multiply by '1' means the Product still remains the same as the Multiplicant value. Multiply by '2' means Left shift the Multiplicand once. Multiply by '3' means addition of multiply by 1and multiply by 2. Multiplier splitter is used to split the multiplier into 2 bit (N/2) sequence which acts as the selection lines of the 4-1 multiplexer. The Partial Product will be chosen based on the selection table given in table 3. Finally the partial product which corresponds to the assigned weight is manipulated from the selection line (B_1 and B_0) by employing a 4 -1 multiplexer.

Table 2 Partial Product Selection table

$\mathbf{B_{N+1}}$	B_N	PARTIAL PRODUCT
0	0	0 X Multiplicand
0	1	1 X Multiplicand
1	0	2 X Multiplicand
1	1	3 X Multiplicand

Wallace Tree Arrangement

Wallace tree has been used in the proposed architecture in order to accelerate multiplication by compressing the number of partial products. This design is done using half adders, Carry save adders and the full adder to speed up the multiplication. In the Wallace tree addition the second partial product had to be shifted left by two bits before adding to the first partial product. Hence the third will be shifted left by four whereas for fourth will be shifted left by six. Hence after proper arrangement all the four partial products will be added.

Sign Detector

The proposed architecture depicted in figure 1 is only able to multiply unsigned numbers. By including the additional block called sign detector to the figure 1 which will able to multiply both signed and unsigned numbers. Sign detector is used to detect the sign bit of the product for the corresponding inputs. Initially MSB of the multiplicand and multiplier is dispatched to the sign detector where the EX-OR operation takes place.

Table 3 Performance Comparison

Parameter	Wallace tree multiplier	Booth multiplier	Splitter based multiplier
Number of Slices	19	18	17
Number of 4 input LUTs	33	32	31
Number of IOs	16	25	23
path delay	21.129ns	17.366ns	11.642 ns

CONCLUSION AND FUTURE ENHANCEMENT

The proposed Splitter Based Multiplier is designed and synthesized to produce the product in a faster manner and also overcomes the limitation of the other basic multipliers. The Synthesis Report shows that the computational time of Splitter Based Multiplier for Unsigned numbers is 11.642 ns and it is lesser than Wallace Tree Multiplier and Booth Multiplier respectively. In the future the proposed SBM can also be implemented using CMOS Technology and the hardware and time requirement of the same can be compared with the existing multipliers. The SBM can also act as MAC unit when it combined with an accumulator.

REFERENCES:

Andrew D. Booth, 'A signed binary multiplication technique. The Quarterly Journal of Mechanics and Applied Mathematics', Volume IV.

Bipin, Ms. Sakshi (2013), 'Design of multiplier using regular partial products', International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 7.

Charan nmKumar .k, S. Vikrama Narasimha Reddy, Neelima Koppala (2013), 'Design of Memory Based Implementation Using LUT Multiplier', International Journal of Engineering Trends and Technology- Volume4,Issue1.

Deepali Chandel, Gagan Kumawat (2013), 'Booth Multiplier: Ease of multiplication', International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 3.

Dakupati.Ravi Sankar, Shaik Ashraf Ali (2013), 'Design of Wallace Tree Multiplier by Sklansky Adder', International Journal of Engineering Research and Applications (IJERA) Vol. 3, Issue 1.

Fadavi-Ardekani.J, ' M x N Booth Encoded Multiplier Generator Using optimized Wallace Trees', IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

Jagadeshwar Rao M, Sanjay Dubey, 'A High Speed Wallace Tree Multiplier Using Modified Booth Algorithm for Fast Arithmetic Circuits", IOSR Journal of Electronics and Communication Engineering (IOSRJECE).

Madrid P.E, B. Millar, and E. E. Swartzlander (1993), 'Modified Boot Algorithm for High Radix Fixed-Point Multiplication', IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

Manuchehr Ebtahimi, 'An ultra low power 57mv Majority function Adder', a thesis submitted at university of water loo.

Meher P.K (2009), 'New Approach to LUT Implementation and Accumulation for Memory-Based Multiplication' IEEE International Symposium on Circuits Systems.

Ohsang Kwon (2002), 'A 16-Bit by 16-Bit MAC Design Using Fast 5:3 Compressor Cells' , Journal of VLSI Signal Processing.

Partha Saray Mohanthy (2009), 'Design and implementation of Faster and Low power Multiplier', the thesis.

Sandeep Shrivastava,Jaikaran Singh (2011), 'Implementation of Radix-2 Booth Multiplier and Comparison with Radix-4 Encoder Booth Multiplier', International Journal on Emerging Technologies.

Sukhmeet Kaur, suman and Manpreet Singh Manna (2013), "Implementation of Modified Booth Algorithm (Radix 4) and its Comparison with Booth Algorithm (Radix-2)", Advance in Electronic and Electric Engineering,

Villeger.D and V. G. Oklobdzija, 'Evaluation of Booth encoding techniques for parallel multiplier implementation', Electronics Letters.

Wallace C.S(1964), 'A suggestion for fast multipliers', IEEE Trans. Electron Comput.,no. EC-13, pp. 14–17