International Journal of Management, IT & Engineering

CONTENTS

Chief Patron

Editorial Board

Reviewers

Title

ANALYSIS, SIMULATION AND COMPARISON OF DIFFERENT MULTIPLIER ALGORITHMS

Authors

Smiksha
M.Tech Student, ECE, UIET
Rohtak, India

Vikas Sindhu
A.P. ECE, UIET
Rohtak, India

Rajender Kumar
A.P, SES, BPSMV
Sonipat, India

Abstract:

High performance systems such as microprocessors, digital signal processors, filters, ALU etc. which is need of hour now days requires a lot of components. One of main component of these high performance systems is multiplier. Most of the DSP computations involve the use of multiply-accumulate operations, and therefore the design of fast and efficient multipliers is imperative. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. This paper investigates analysis of different multiplier for speed, area and memory usage. we try to present an efficient multiplier is produce fast, accurate and require minimum area. In this paper we will first study three different types of multipliers: array multiplier, radix-2 and radix-4 modified booth multiplier algorithm. Then we compared the working of different multipliers by comparing the memory usage, speed and area by each of them. The result of this paper helps us to choose a better option to choose a better multiplier out of three multipliers in fabricating different systems.

Keywords: Multiplier, speed, area, DSP, System.

Introduction:

Multiplication is one of the basic arithmetic operations. Digital multiplication is not the most fundamentally complex operation, but is the most extensively used operation.

The function of a binary unsigned multiplier, like its decimal counterpart, consists of a multiplicand (X), a multiplier (Y), and a product (P). The result is the product of the multiplier and the multiplicand (P = X * Y). Multiplication is not simple as addition or

Subtraction operations, because it takes more time to perform two subtasks, addition and shifting. Until the late 1970s, most microcomputers did not have a multiply instruction, so programmers used a ‘multiply routine’, which repeatedly shifts and accumulates the partial result. This method is very easy but it has two big disadvantages.

First, this method is very time consuming as it takes many steps to complete the multiplication operation. Second one is sign problem. The basic school method handles the sign

with a separate rule. To solve these problems different types of multiplier were introduced. These multipliers are used in different applications of digital electronic as digital signal processing, microprocessor, filters etc. In digital signal processing system multipliers plays a main role. Multiplier can be serial or parallel [5]. The serial multiplier uses the basic shift and adds operation whereas in the parallel multipliers partial products are parallel added. In this paper we are comparing the three different multipliers- array multiplier, radix2 multiplier, radix 4 multiplier.

Different multiplier:

As we know in multiplication operation there are two operands, one is multiplicand and other is multiplier. In binary number system we do multiplication by using different type of multiplier. A binary multiplier [6] uses the simple shift and adds operation. There are many multipliers introduced in digital electronics. Some of them are given below:

Array multiplier:

An array multiplier is a parallel multiplier [4] which does shift and adds all at once. This multiplier is called an array because it has array of adders. An array multiplier also uses shift and adds operation as in binary multiplier but it adds the partial products parallel [7][3]. The following figure shows the 4x4 array multiplier.

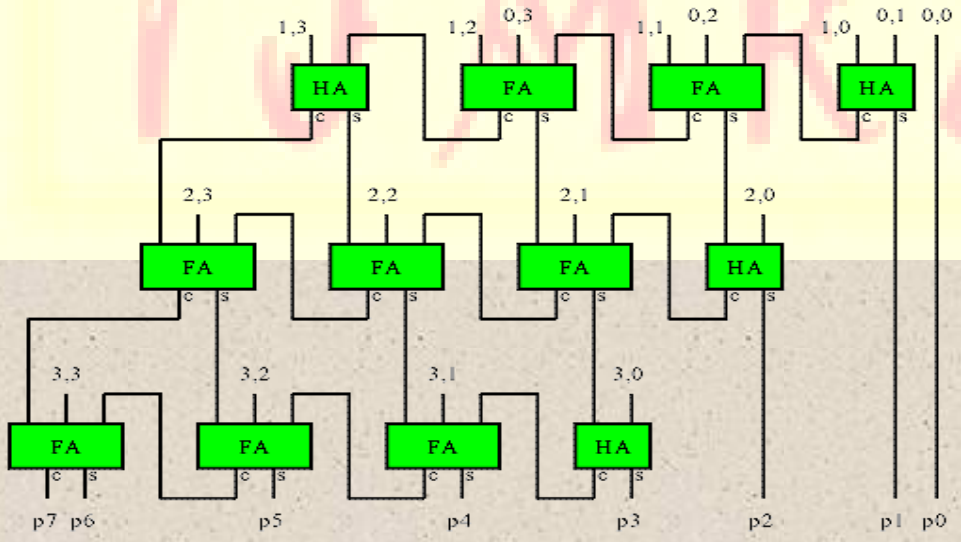


Fig1: 4x4 array multiplier

This can be explained as:

A and B are two four bit numbers as shown in the figure1.1

A= a_3 a_2 a_1 a_0 multiplicand

B= b_3 b_2 b_1 b_0 multiplier

Prod = p7 p6 p5 p4 p3 p2 p1 p0 product

An array multiplier adds the partial product parallel. This multiplier uses half adder and full adder on the basis of no. of bits in multiplier and multiplicand. More no. of bits requires more no. of adders. A general hardware requirement in a array multiplier can be given as:

Let 'm' is the no. of bits in multiplicand and 'n' is the no of bits in multiplier than no. of half adder required is same as no. of bits in multiplier i.e. n, and no. of full adder should be (m-2)n.

This multiplier takes more time to give the product because it takes time in processing carry and sum. If 'Tc' is carry propagation time, 'Ta' is AND gate time and 'Ts' is sum propagation time then multiplication time can be given as

When Tc>Ts then

Multiplication time is Ta + ((n-1)(m-1))Tc

And when Tc<Ts then

Multiplication time is Ta + (n-1)Ts + (m-1)Tc.

So this multiplier has simple operation but very time consuming, so it will make the system slow.

Radix 2^n multiplier:

Radix 2^n multiplier was introduced by M.K. Ibrahim in 1983[2]. The architecture of a radix 2 n multiplier is given in the Figure2. This diagram shows the multiplication of two numbers U and V with four digits each.

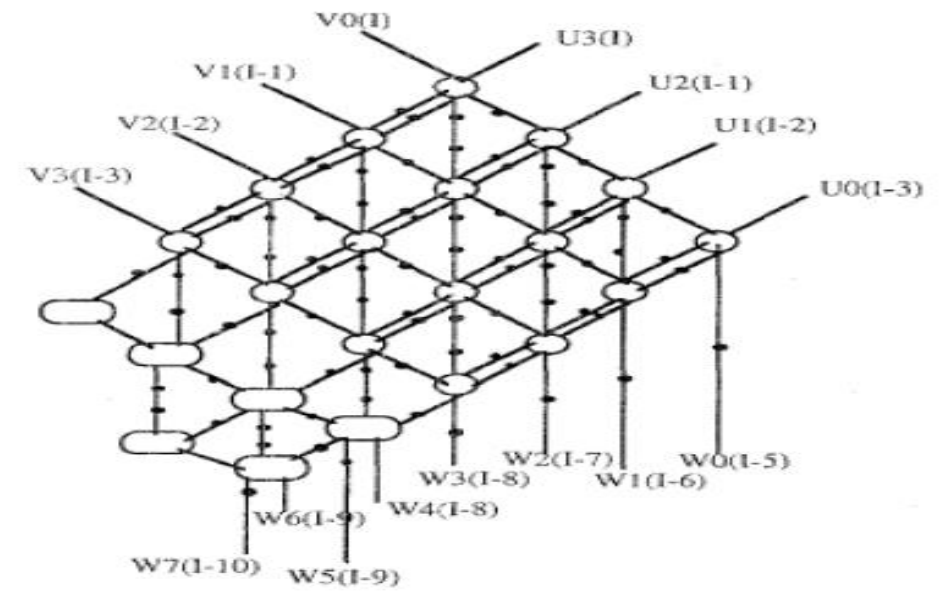


Fig 2: Radix 2n architecture

Each circle in this diagram shows a radix cell which is the heart of the design. Every radix cell has four digit inputs and two digit outputs. The input digits are also fed through the corresponding cells. The dots in the figure represent latches for pipelining. Every dot consists of four latches. The ellipses represent adders which are included to calculate the higher order bits. The outputs are shown by W's. The 1's denote the clock period at which the data appear.

Booth algorithm for radix 2 multiplier:

Booth algorithm gives a procedure for multiplying binary integers in signed –2's complement representation [8]. The Booth's algorithm serves two purposes:

1. Fast multiplication (when there are consecutive 0's or 1's in the multiplier).

2. Signed multiplicatio Booth algorithm can be explained with the following steps: let us take an example: 2 x (- 4) i.e. 0010 x 1100

Step 1: Making the Booth table

I. From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.

i.e., 0010 -- From 0 to 0 no change, 0 to 1 one change, 1 to 0 another change, and so there are two changes on this one.

1100 -- From 1 to 1 no change, 1 to 0 one change, 0 to 0 no change, so there is only one change on this one.

Therefore, multiplication of 2 x (– 4) where 2(0010) is the multiplicand and (– 4) i.e. (1100) is the multiplier.

II. Let X = 1100 (multiplier) and Y = 0010 (multiplicand)

Take the 2's complement of Y and call it –Y

–Y = 1110

III. Load the X value in the table.

IV. Load 0 for X-1 value it should be the previous first least significant bit of X

V. Load 0 in U and V rows which will have the product of X and Y at the end of operation.

VI. Make four rows for each cycle; this is because we are multiplying four bits numbers.

Table 1: booth algorithm

U	V	X	X-1	Load the value
0000	0000	1100	0	1st cycle
				2nd cycle
				3rd cycle
				4thcycle

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:

Look at the first least significant bits of the multiplier "X", and the previous least significant bits of the multiplier "X - 1".

I. 0 0-- Shift only

1 1--Shift only.

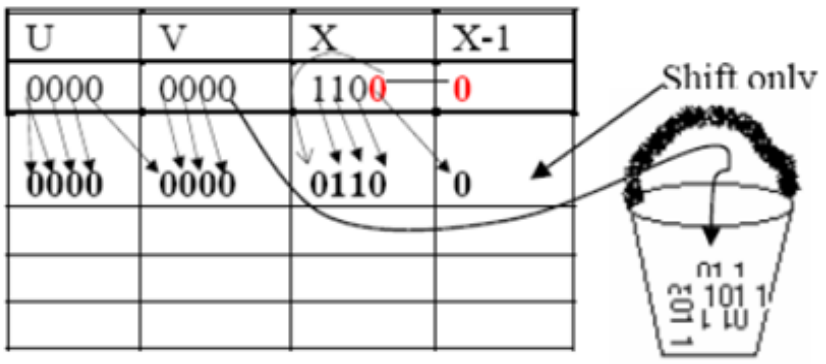
0 1--Add Y to U, and shift

1 0--Subtract Y from U, and shift or add (-Y) to U and shift

II. Take U & V together and shift arithmetic right shift which preserves the sign bit of 2's complement number. Thus a positive number remains positive, and a negative number remains negative.

III. Shift X circular right shifts because this will prevent us from using two registers for the X value.

Table.2: Booth algorithm



Repeat the same steps until the four cycles complete as shown in the next table

Table 3: Booth algorithms

U	V	X	X-1	
0000	0000	1100	0	
0000	0000	0110	0	
0000	**0000**	**0011**	**0**	← Shift only

Table 4: Booth algorithm

U	V	X	X-1	
0000	0000	1100	0	
0000	0000	0110	0	
0000	0000	0011	0	
1110	**0000**	**0011**	**0**	← Add –Y (0000 + 1110 = 1110)
1111	**0000**	**1001**	**1**	← Shift

Table 5: Booth algorithm

U	V	X	X-1	
0000	0000	*1100*	0	
0000	0000	0110	0	
0000	0000	0011	0	
1110	0000	0011	0	
1111	0000	1001	1	
1111	**1000**	***1100***	**1**	← Shift only

We have finished four cycles, so the answer is shown in the last rows of U and V which is 11111000

Booth algorithm for radix 4 multiplier:

For a Radix^4 booth multiplier firstly we have to study the booth encoder. To avoid variable size partial product arrays we will use modified booth encoder. Firstly, to covert a number in to radix4 number a zero is appended into the Least Significant Bit (LSB) of the multiplier, then dividing them into three digits respectively according to Booth Encoder Table1.6. In an 8 bit multiplier four partial products will be generated using booth multiplier approach instead of eight partial products being generated using conventional multiplier.

Table.6: Modified Booth Encoder's table to generate radix 4 number

X(i)	X(i-1)	X(i-2)	y	Partial Product
0	0	0	0	0
0	0	1	1	1x multiplicand
0	1	0	1	1x multiplicand
0	1	1	2	2x multiplicand
1	0	0	-2	-2x multiplicand
1	0	1	-1	-1x multiplicand
1	1	0	-1	-1x multiplicand
1	1	1	0	0

Table 1.6 shows the Radix4 modified booth algorithm scheme for odd values of 'i'. X(i), X(i-1) and X(i-2) are three bits wide binary numbers of the multiplier. X(i) is the most significant bit (MSB) and X(i-2) is the least significant bit (LSB), y is representing the Radix-4 number of the 3-bit binary multiplier number.

Simulation results and comparison of different multiplier:

Simulation of different multiplier is done using Xilinx 9.1i software and results are shown. The device utilization summary is shown in table 7 which shows that radix-2 booth multiplier required lesser time and memory area in designing system.

Table 7: Device utilization Summary for Different multiplier

S. No.	Type of Multiplier	Device Utilization Summary			
		No. of Slices	No. of IOBs	Timing (in ns)	Memory usage (in kilobytes)
1	Array multiplier	71/1920	32/173	32.001	148492
2	Radix 2 booth multiplier	0/1920	33/173	6.445	143372
3	Radix 4 booth multiplier	19/1920	27/173	16.230	144396

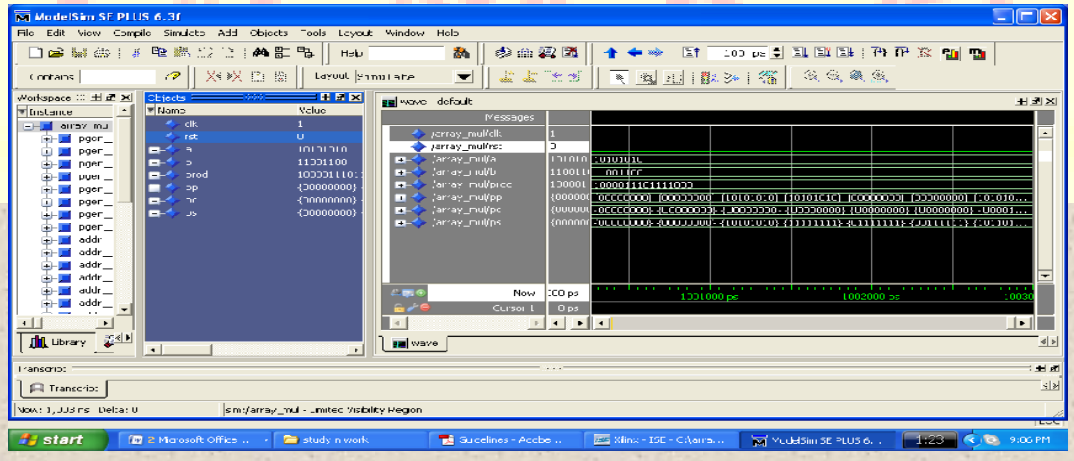


Fig.3. Simulation result of array multiplier

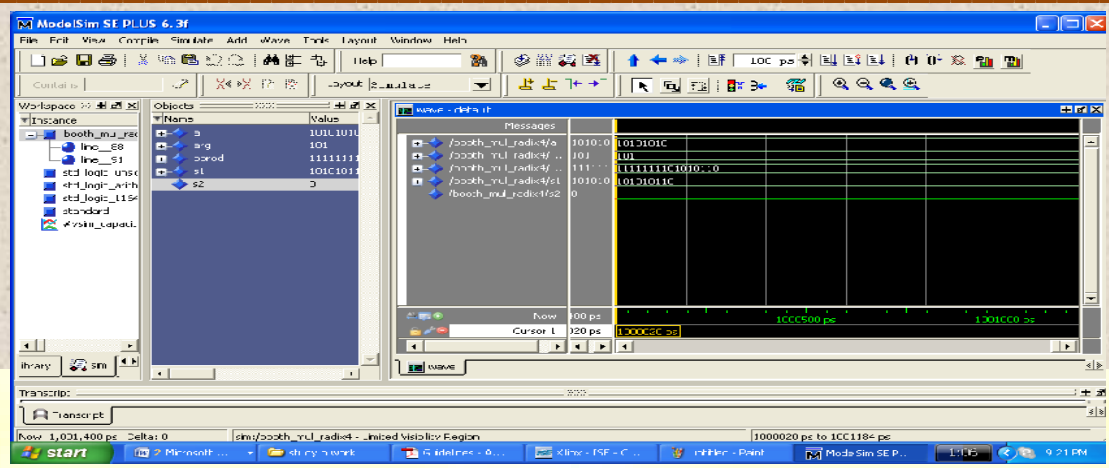


Fig.4 Simulation result of radix 2 booth multiplier

Conclusion:

As multiplication is most critical and most frequently used operation in many systems such DSP, filter and processors. So we always need to find a better solution in case of multipliers. Our multipliers should always consume less area and cover less timing. This paper gives a clear comparison of three multipliers. While comparing the array multiplier, radix 2 and the radix 4 booth multipliers we found that the array multipliers have the maximum area. This is because it uses a large number of adders. As a result it slows down the system. And radix 2 multiplier is the better multiplier than others. This is because it uses less memory, takes less timing as compared with others. So through our paper we try to determine which of the three multiplier algorithms works the best. In the end, we determine that radix 2 modified booth algorithm works the best.

References:

- Yijun Liu, Steve Furber: The Design of a Low Power Asynchronous Multiplier, The Advanced Processor Technologies Group, The Department of Computer Science, The University of Manchester, Manchester 2004, M13 9PL, UK
- M.K. Ibrahim, PhD: Radix-2" multiplier structures : a structured design methodology, Department of Electrical and Electronic Engineering, University of Nottingham, University Park, Nottingham, NG7 2RD, United Kingdom IEE PROCEEDINGS-E, Vol. 140, No. 4, JULY 1993

- Saeid Gorgin, Ghassem Jaberipur, Behrooz Parhami: Design and Evaluation of Decimal Array Multipliers, Dept. of Electrical & Computer Engr., Shahid Beheshti Univ., Tehran, Iran, School of Computer Sci., Inst. for Research in Fundamental Sci. (IPM), Tehran, Iran, Dept. of Electrical & Computer Engr. Univ. of California, Santa Barbara, USA. 2009 IEEE
- Ma Lin1,Gao2,Yan3: Instruction Level Test for Parallel Multipliers, Key Laboratory of Computer System and Architecture, 1Institute of Computing Technology, Chinese Academy of Sciences, 2College of Electronics and Information Engineering, Tongji University,3Graduate University of Chinese Academy of Science, Beijing China, 100049,2008 IEEE
- Danny Crookes, Richard M. Jiang: A Low-Power High-Radix Serial-Parallel Multiplier, The School of Electrical Engineering, Electronics & Computer Science, Queen's University Belfast, Belfast BT7 1NN, UK, 2007 IEEE
- R. GNANASEKARAN: A Fast Serial-Parallel Binary Multiplier, IEEE TRANSACTIONS ON COMPUTERS, VOL. c-34, NO. 8, AUGUST 1985
- JOHN PAUL SHEN AND F. JOEL FERGUSON: The Design of Easily Testable VLSI Array Multipliers, Center for Computer-Aided Design, Department of Electrical and Computer Engineering, Carnegie-Mellon University, IEEE TRANSACTIONS ON COMPUTERS, VOL. c-33, NO. 6, JUNE 1984
- Leonard0 L.de Oliveira Eduardo Costa, Sergio Bampi, JosC Monteiro: Array Hybrid Multiplier versus Modified Booth Multiplier: Comparing Area and Power Consumption of Layout Implementations of Signed Radix-4 Architectures, 2004 IEEE.
- Yun-Nan Chang , Janardhan H. Satyanarayana, Keshab K. Parhi :Systematic Design of High-Speed and Low-Power Digit-Serial Multipliers ,Student Member, IEEE, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING, VOL. 45, NO. 12, DECEMBER 1998.
- www.wikipedia.com
- www.xilinx.com
- www.andraka.com/multipli.htm
- Circuit Design using VHDL, by Pedroni
- VHDL by B. Bhaskar