International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

A Review on Performance Evaluation of Different Digital Multipliers in VLSI using VHDL

Bhawna Singroul
M. Tech Student
Department of Electronics & Communication Engineering
Dr. C.V. Raman University Kota, Bilaspur
C.G., India

Pallavee Jaiswal
Assistant Professor
Department of Electronics & Communication Engineering
Dr. C.V. Raman University Kota, Bilaspur C.G., India

Abstract **The importance of digital electronics is increasing day-by-day in our day-today life. Digital multipliers have great importance in designing modern gadgets, in digital signal processing and in many other applications. For the improvement in performance of modern devices and softwares, there is a need of designing a multiplier having high speed, less area, low power consumption, simple and regular design. Here is a review presented by comparing performance of various multipliers. The multipliers are mostly compared in terms of delay power consumption and area required etc. parameters. This paper brings all important digital multipliers together for comparative analysis. This comparative analysis helps us to select one most suitable multiplier for a particular application.**

Keywords: Digital multipliers, comparison, delay, power, area, vedic, wallce, booth.

1. INTRODUCTION

1.1 Overview

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. Multipliers are key components of many high performance systems such as FIR filters, 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 element in the system. Furthermore, it is generally the most area consuming. 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, whole spectrums of multipliers with different area-speed constraints have 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 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. 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.

1.2 Filter

A filter is a device or process that removes some unwanted components or features from a signal. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal. Most often, this means removing some frequencies or frequency bands. However, filters do not exclusively act in the frequency domain; especially in the field of image processing many other targets for filtering exist. Correlations can be removed for certain frequency components and not for others without having to act in the frequency domain. Filters are widely used in electronics and telecommunication, in radio, television, audio recording, radar, control systems, music synthesis, image processing, and computer graphics.Filters are circuits which perform signal processing functions, specifically to remove unwanted frequency components from the signal, to enhance wanted ones, or both.

Digital filters are very important part of DSP. Infect their extraordinary performance is one of the key reasons that DSP has become so popular. Filters have two uses: signal separation and signal restoration. Signal separation is needed when the signal has been contaminated with interference, noise or other signals. For example imagine a device for measuring the electrical activity of a baby's heart (EKG) while in the womb. The raw signal will be likely to be corrupted by the breathing and the heartbeat of the mother. A filter must be used to separate these signals so that they can be individually analysed.

There are many different bases of classifying filters and these overlap in many different ways; there is no simple hierarchical classification. Filters may be:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

1. Passive or active
2. Analog or digital
3. High-pass, low-pass, band-pass, band-stop (band-rejection; notch), or all-pass.
4. Discrete-time (sampled) or continuous-time
5. Linear or non-linear

1.2.1 Infinite impulse response (IIR)

Infinite impulse response (IIR) is a property applying to many linear time-invariant systems. Common examples of linear time-invariant systems are most electronic and digital filters. Systems with this property are known as IIR systems or IIR filters, and are distinguished by having an impulse response which does not become exactly zero past a certain point, but continues indefinitely. This is in contrast to a finite impulse response (FIR) in which the impulse response h (t) does become exactly zero at times t > T for some finite T, thus being of finite duration.

1.2.2 Finite Impulse Response filters (FIR)

Finite Impulse Response filters are the most important element in signal processing and communication. FIR filter architecture has multiplier, adder and delay unit. So FIR filter performance is mainly based on multiplier.

An FIR filter has a number of useful properties which sometimes make it preferable to an infinite impulse response (IIR) filter. FIR filters:

1. Require no feedback. This means that any rounding errors are not compounded by summed iterations. The same relative error occurs in each calculation. This also makes implementation simpler.
2. Are inherently stable, since the output is a sum of a finite number of finite multiples of the input values, so can be no greater than $\sum \quad i$ times the largest value appearing in the input.
3. Can easily be designed to be linear phase by making the coefficient sequence symmetric. This property is sometimes desired for phase-sensitive applications, for example data communications, seismology, crossover filters, and mastering.

The main disadvantage of FIR filters is that considerably more computation power in a general purpose processor is required compared to an IIR filter with similar sharpness or selectivity, especially when low frequency (relative to the sample rate) cutoffs are needed. However many digital signal processors provide specialized hardware features to make FIR filters approximately as efficient as IIR for many applications.

FIR filters are widely used because they have linear phase characteristics, guarantee stability and are easy to implement with multipliers, adders and delay elements. The number of taps in digital filters varies according to applications. In commercial filter chips with the fixed number of taps zero coefficients are loaded to registers for unused taps and unnecessary calculations have to be performed. To alleviate this problem, the FIR filter chips providing variable-length taps have been widely used in many application fields. However, these FIR filter chips use memory, an address generation unit, and a modulo unit to access memory in a circular manner. The paper proposes two special features called a data reuse structure and a recurrent-coefficient scheme to provide variable-length taps efficiently. Since the proposed architecture only requires several MUXs, registers, and a feedback-loop, the number of gates can be reduced over 20 % than existing chips.

In, general, FIR filtering is described by a simple convolution operation as expressed in the equation (1)

$$Y[n] = \sum_{k=0}^{N-1} \quad h[k]x[n - k] \quad (1)$$

where x[n], y[n], and h[n] represent data input, filtering output, and a coefficient, respectively and N is the filter order. The equation using the bit-serial algorithm for a FIR filter can be represented as

$$y(n) = \sum_{k=0}^{N-1} \quad \sum_{j=0}^{M-1} \quad (h_f[k].2^j).x[n-k] \quad (2)$$

Where, theh_f, N and M are the jth bit of the coefficient.

1.3 Multiplier

A multiplier is one of the key hardware blocks in most digital signalprocessing (DSP) systems. Typical DSP applications where a multiplier plays animportant role include digital filtering, digital communications and spectralanalysis. Many current DSP applications are targeted atportable, battery-operated systems, so that power dissipation becomes one of theprimary design constraints. Since multipliers are rather complex circuits and musttypically operate at a high system clock rate, reducing the delay of a multiplier isan essential part of satisfying the overall design.

1.4 Types of multiplier

1.4.1 Binary multiplier

A Binary multiplier is an electronic hardware device used in digital electronics or a computer or other electronic device to perform rapid multiplication of two numbers in binary representation. It is built using binary adders.The rules for binary multiplication can be stated as follows

1. If the multiplier digit is a 1, the multiplicand is simply copied down and represents the product.

2. If the multiplier digit is a 0 the product is also 0.

For designing a multiplier circuit we should have circuitry to provide or do the following three things:

1. It should be capable identifying whether a bit is 0 or 1.

2. It should be capable of shifting left partial products.

3. It should be able to add all the partial products to give the products as sum of partial products.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

4. It should examine the sign bits. If they are alike, the sign of the product will be a positive, if the sign bits are opposite product will be negative. The sign bit of the product stored with above criteria should be displayed along with the product.

From the above discussion we observe that it is not necessary to wait until all the partial products have been formed before summing them. In fact the addition of partial product can be carried out as soon as the partial product is formed.

1.4.2 Baugh Wooley Multiplier

Baugh-Wooley algorithm for the unsigned binary multiplication is based on the concept shown in figure. The algorithm specifies that all possibleAND terms are created first, and then sent through an array of half-adders andfull-adders with the Carry-outs chained to the next most significant bit at each level of addition. Negative operands may be multiplied using a Baugh-Wooleymultiplier.

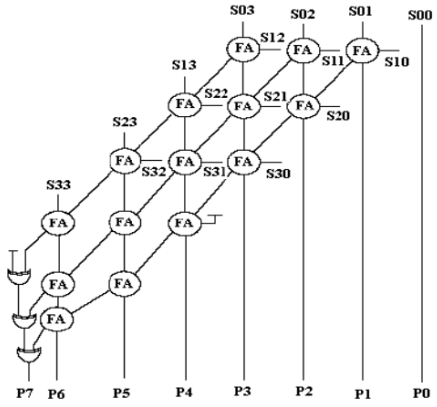

Figure 1.2: Baugh Wooley multiplier architecture

1.4.3 Braun multiplier

The simplest parallel multiplier is the Braun array. All the partial products are computed in parallel, then collected through a cascade of Carry Save Adders. The completion time is limited by the depth of the carry save array, and by the carry propagation in the adder. Note that this multiplier is only suited for positive operands. The structure of the Braun algorithm for the unsigned binary multiplication is shown in figure

Figure 1.3: Braun multiplier architecture

1.4.4 Wallace multiplier

The partial-sum adders can also be rearranged in a tree like fashion, reducing both the critical path and the number of adder cells needed. The presented structure is called the Wallace tree multiplier and its implementation is shown in figure. The tree multiplier realizes substantial hardware savings for larger multipliers. The propagation delay is reduced as well. In fact, it can be shown that the propagation delay through the tree is equal to O (log3/2 (N)). While substantially faster than the carry-save structure for large multiplier word lengths, the Wallace multiplier has the disadvantage of being vary irregular, which complicates the task of an efficient layout design.

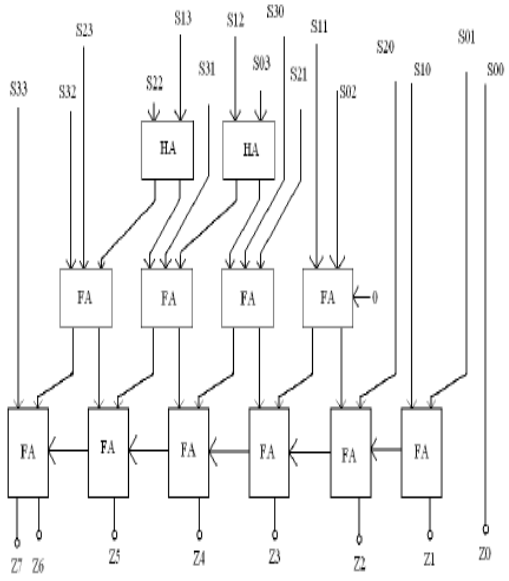

Figure 1.4: Wallace multiplier architecture

1.4.5 Combinational Multiplier

Combinational Multipliers do multiplication of two unsigned binary numbers .This multiplier is also used for the multiplication of two signed number. Each bit of the multiplier is multiplied against the multiplicand, the product is associated according to the position of the bit within the multiplier, and the resulting products are then added to form the final result. Main advantage of binary multiplication is that the generation of intermediate products are easy. If the multiplier bit is a 1, the product is an correctly shifted copy of the multiplicand; if the multiplier bit is a 0, the product is simply 0.In most of the systems combinational multipliers are slow and take a lot of area. [15]

1.4.6 Array Multiplier

Array multiplier is well known due to its regularstructure. Multiplier circuit is based on repeated additionand shifting procedure. Each partial product is generatedby the multiplication of the multiplicand with onemultiplier digit. The partial product are shifted accordingto their bit sequences and then added. The summation canbe performed with normal carry propagation adder. N-1adders are required where N is the no. of multiplier bits [15]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

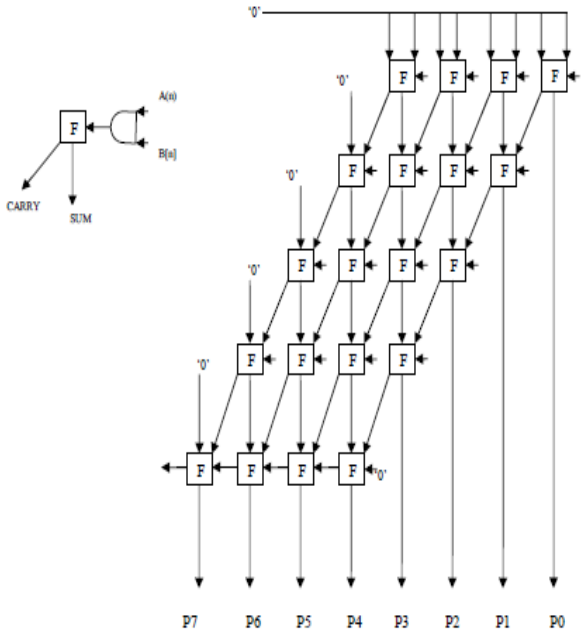

Figure 1.5: Array multiplier architecture

1.4.7 Sequential Multiplier

If we want to multiply two binary number (multiplicand X has n bits and multiplier Y has m bits) using single n bit adder, we can built a sequential circuit that processes a single partial product at a time and then cycle the circuit m times. This type of circuit is called sequential multiplier. Sequential multipliers are attractive for their low area requirement. In a sequential multiplier, the multiplication process is divided into some sequential steps. In each step some partial products will be generated, added to an accumulated partial sum and partial sum will be shifted to align the accumulated sum with partial product of next steps. Therefore, each step of a sequential multiplication consists of three different operations which are generating partial products, adding the generated partial products to the accumulated partial sum and shifting the partial sum. [15]

1.4.8 Booth multiplier

The decision to use a Radix-4 modified Booth algorithm rather than Radix-2 Booth algorithm is that in Radix-4, the number of partial products is reduced to n/2. Though Wallace Tree structure multipliers could be used but in this format, the multiplier array becomes very large and requires large numbers of logic gates and interconnecting wires which makes the chip design large and slows down the operating speed.

The Booth multiplier makes use of Booth encoding algorithm in order to reduce the number of partial products by processing three bits at a time during recoding .This recoding method is widely used to generate the partial products for implementation of large parallel multipliers, which adopts the parallel encoding scheme.

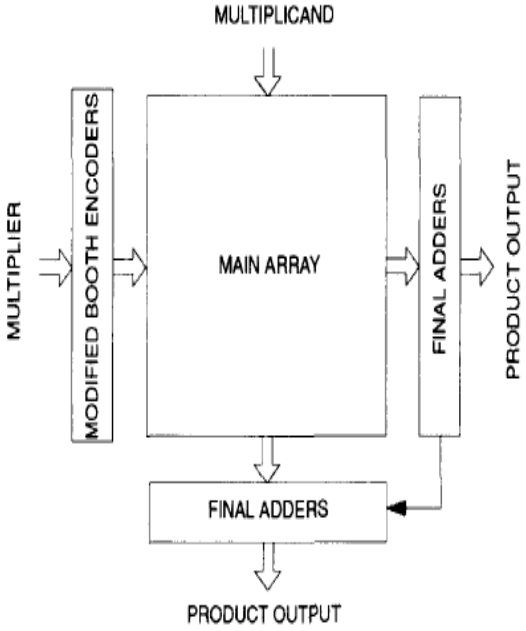

Figure 1.6: Block diagram of Booth multiplier

1.5 Radix multiplication algorithm

Radix multiplication algorithms can beused to design iterative array multipliers, serial/parallel multipliers and serialmultipliers. Due to the iterative nature of the algorithms, the resulting structuresare regular, modular and require localized communication only, which makes themsuitable for VLSI implementation. The advantage of the structures based on theradix approach is that the architecture of the basic cell is not fixed for all radices.Any architecture can be used so long as its functionality satisfies the correspondingradix multiplication algorithm. The new algorithms can be used as a structuredmultiplier-design methodology that will allow designers to find the best compromisebetween hardware cost and multiplication time. The multiplier architecture is first defined in terms of the radix-2^nmultiplication algorithm which is general for all n. This architecture being available for every n. The trade-off between cost and time is then achieved by choosing the radix that gives the best performance.

1.5.1 Radix 2^n multiplication algorithm

The architecture of a radix 2^n multiplier is given in the Figure. This block diagram shows the multiplication of two numbers with four digits each. These numbers are denoted as V and U while the digit size was chosen as four bits. The reason for this will become apparent in the following sections. Each circle in the figure corresponds to 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

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

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. They do not fit the regularity of the design as they are used to "terminate" the design at the boundary. The outputs are again in terms of four bit digits and are shown by W's. The 1's denote the clock period at which the data appear.

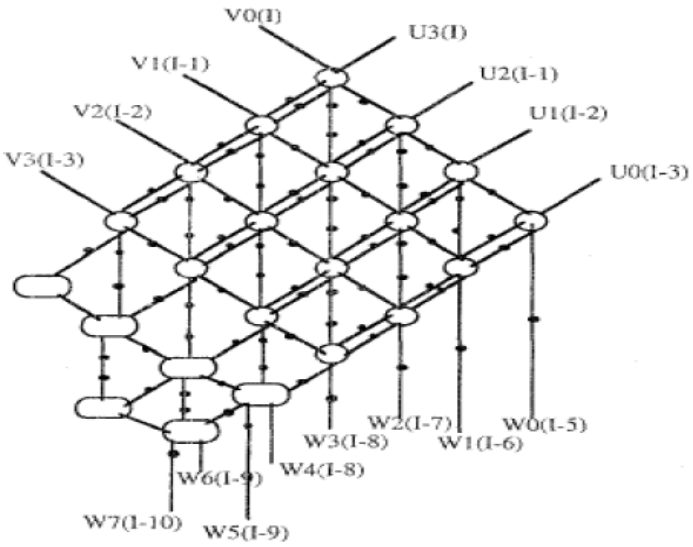

Figure 1.7: Radix 2^n multiplier architecture

2. METHODOLOGY

2.1 Overview

In this project we first designed three different types of multipliers using shift and add method, radix 2 and radix 4 modified booth multiplier algorithm. We used different type of adders like sixteen bit full adder in designing that multiplier. Then we designed a 4 tap delay FIR filter and in place of the multiplication and additions we implemented the components of different multipliers and adders. Then we compared the working of different multipliers by comparing the power consumption by each of them.

2.2 VHDL: The language

An entity declaration, or entity, combined with architecture or body constitutes a VHDL model. VHDL calls the entity-architecture pair a design entity. By describing alternative architectures for an entity, we can configure a VHDL model for a specific level of investigation. The entity contains the interface description common to the alternative architectures. It communicates with other entities and the environment through ports and generics. Generic information particularizes an entity by specifying environment constants such as register size or delay value. For example,

entity**A is**
port (x, y: in real; **z:** out real);
generic (delay: time);
end**A;**

The architecture contains declarative and statement sections. Declarations form theregion before the reserved word **begin** and can declare local elements such as signals and components. Statements appear after *begin* and can contain concurrent statements. For instance,

architecture B of **A** is
component M
port (j : in real ; **k** : out real);
end component;
signala,b,c real := 0.0;
begin
"concurrent statements"
end B;

The variety of concurrent statement types gives VHDL the descriptive power to createand combine models at the structural, dataflow, and behavioural levels into one simulation model. The structural type of description makes use of component instantiation statements to invoke models described elsewhere. After declaring components, we use them in the component instantiation statement, assigning ports to local signals or other ports and giving values to generics invert: M port map (j => a ; **k** =>***c);*** We can then bind the components to other design entities through configuration specifications in VHDL's architecture declarative section or through separate configuration declarations. The dataflow style makes wide use of a number of types of concurrent signal assignment statements, which associate a target signal with an expression and a delay. The list of signals appearing in the expression is the sensitivity list; the expression must be evaluated for any change on any of these signals. The target signals obtain new values after the delay specified in the signal assignment statement. If no delay is specified, the signal assignment occurs during the next simulation cycle:
c <= a + b after delay;
VHDL also includes conditional and selected signal assignment statements. It uses block statements to group signal assignment statements and makes them synchronous with a guarded condition. Block statements can also contain ports and generics to provide more modularity in the descriptions. We commonly use concurrent process statements when we wish to describe hardware at the behavioural level of abstraction. The process statement consists of declarations and procedural types of statements that make up the sequential program. Wait and assert statements add to the descriptive power of the process statements for modelling concurrent actions:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

```
Process
begin
variable i : real := 1.0;
wait on a;
   i = b * 3.0;
   c <= i after delay;
end process;
```

Other concurrent statements include the concurrent assertion statement, concurrentprocedure call, and generate statement. Packages are design units that permit types and objects to be shared. Arithmetic operations dominate the execution time of most Digital Signal Processing (DSP) algorithms and currently the time it takes to execute a multiplication operation is still the dominating factor in determining the instruction cycle time of a DSP chip and Reduced Instruction Set Computers (RISC). Among the many methods of implementing high speed parallel multipliers, there is one basic approach namely Booth algorithm.

Power consumption in VLSI DSPs has gained special attention due to the proliferation of high-performance portable battery-powered electronic devices such as cellular phones, laptop computers, etc. DSP applications require high computational speed and, at the same time, suffer from stringent power dissipation constraints.Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. Among these, the Wallace multiplier is among thefastest. However, they suffer from a bad regularity. Hence, when regularity, high performance and low power are primary concerns, Booth multipliers tend to be theprimary choice.Booth multipliers allow the operation on signed operands in 2's complement. They derive from array multipliers where, for each bit in a partial product line, an encoding scheme is used to determine if this bit is positive, negative or zero. The Modified Booth algorithm achieves a major performance improvement through radix-4 encoding. In this algorithm each partial product line operates on 2 bits at a time, thereby reducing the total number of the partial products. This is particularly true for operands using 16 bits or more.

2.3 Shift and add multiplier

Shift-and-add multiplication is similar to the multiplication performed by paper and pencil. This method adds the multiplicand X to itself Y times, where Y denotes the multiplier. To multiply two numbers by paper and pencil, the algorithm is to take the digits of the multiplier one at a time from right to left, multiplying the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

As an example, consider the multiplication of two unsigned 4-bit numbers, 8 (1000) and 9 (1001).

```
Multiplicand 1000 ×
Multiplier1001
                    ----------
 1000
 0000
0000
 1000
                   ----------------
Product1001000
```

In the case of binary multiplication, since the digits are 0 and 1, each step of the multiplication is simple. If the multiplier digit is 1, a copy of the multiplicand (1 × multiplicand) is placed in the proper positions; if the multiplier digit is 0, a number of 0 digits (0 × multiplicand) are placed in the proper positions.

2.4 Booth multiplier (Radix2)

Booth's algorithm is based upon recoding the multiplier, y, to a recoded, value, z, leaving the multiplicand, x, unchanged. In Booth recoding, each digit of the multiplier can assume negative as well as positive and zero values. There is a special notation, called signed digit (SD) encoding, to express these signed digits. In SD encoding +1 and 0 are expressed as 1and 0, but -1 is expressed as 1

The value of a 2s complement integer was defined a by equation.

This equation says that in order to get the value of a signed 2's complement number, multiply the m – ith digit by -2`-1, and multiply each remaining digit i by +2g.

For example, -7, which is 1001 in 2's complement notation, would be, in SD notation, 1001 = -8 + 0 + 0 + 1 = -7.

For implementing booth algorithm most important step is booth recoding. By booth recoding we can replace string of 1s by 0s. For example the value of strings of five 1s, 11111 = 2^5 – 1 = 100001 = 32 – 1 = 31. Hence if this number were to be used as the multiplier in a multiplication, we could replace five additions by one addition and one subtraction.

The Booth recoding procedure, then, is as follows:

1. Working from lsb to msb, replace each 0 digit of the original number with a 0 in the recoded number until a 1 is encountered.

2. When a 1 is encountered, insert a 1 at that position in the recoded number, and skip over any succeeding I's until a 0 is encountered.

3. Replace that 0 with a 1 and continue.

This algorithm is expressed in tabular form in Table 3.1, considering pairs of

numbers, y_{i-1} and y_i, and the recoded digit, z_i, shown in Table 3.1

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

y_i	y_{i-1}	z_{i-i}	Multiplier Value	Situation
0	0	0	0	String of 0s
0	1	1	+1	End of string of 1s
1	0	1	-1	Begin string of 1s
1	1	0	0	String of 1s

Table 2.1: Booth recoding table for radix-.

2.5 Booth Multiplication Algorithm for radix 2

Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation. I will illustrate the booth algorithm with the following example:

Example, 2 ten x (- 4) ten

0010 two* 1100 two

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 ten (0010 two) is the multiplicand and (– 4) ten (1100 two) is the multiplier.

II. Let X = 1100 (multiplier)

Let 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.

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

Table 2.2: Making the Booth table

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 shift because this will prevent us from using two registers for the X value.

U	V	X	X-1
0000	0000	1100	0
0000	0000	0110	0

Shift only

Table 2.3: I 0 0 Shift only

Repeat the same steps until the four cycles are completed

U	V	X	X-1
0000	0000	1100	0
0000	0000	0110	0
0000	0000	0011	0

Shift only

Table 2.4: 1 1 Shift only.

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

Table 2.5: 0 1 Add Y to U, and shiftShift

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

Add – Y (0000 + 1110 – 1110)

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

Table 2.6: 1 0 Subtract Y from U, and shift or add (-Y) to U and shift

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

Note: By the fourth cycle, the two algorithms have the same values in the product register

2.6 Modified booth multiplier (Radix 4)

The Booth multiplier makes use of Booth encoding algorithm in order to reduce the number of partial products by processing three bits at a time during recoding. This recoding method iswidely used to generate the partialproducts for implementation of large parallel multipliers, which adopts the parallel encoding scheme.

2.7 Booth multiplication algorithm for radix 4

One of the solutions of realizing high speed multipliers is to enhance parallelism which helps to decrease the number of subsequent calculation stages. The original version of the Booth algorithm (Radix-2) had two drawbacks. They are: (i) the number of add subtract operations and the number of shift operations become variable and become inconvenient in designing parallel multipliers. (ii) The algorithm becomes inefficient when there are isolated 1's. These problems are overcome by using modified Radix4 Booth algorithm which scans strings of three bits with the algorithm given below:

1) Extend the sign bit 1 position if necessary to ensure that n is even.

2) Append a 0 to the right of the LSB of the multiplier.

3) According to the value of each vector, each Partial Product will he 0, +y ,-y, +2y or -2y.

The negative values of y are made by taking the 2's complement and in this paperCarry-look-ahead (CLA) fast adders are used. The multiplication of y is done by shifting y by one bit to the left. Thus, in any case, in designing a n-bit parallel multipliers, only n/2 partial products are generated.

X(i)	X(i-1)	X(i-2)	Y
0	0	0	+0
0	0	1	+y
0	1	0	+y
0	1	1	+2y
1	0	0	-2y
1	0	1	-y
1	1	0	-y
1	1	1	+0

Table 2.7: Radix4 Modified Booth algorithm scheme for odd values of i.

3 REFERENCES

[1] R. Jaikumar, P. Poongodi and R. Lavanya," Implementation of high speed arithmetic logic using vedic mathematics techniques" ictact journal on microelectronics, february 2015

[2] M. Ramalatha, K. Deena, Dayalan ,Dharani "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques" ACTEA IEEE 2009.

[3] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Design of fixed width multipliers with linear compensation function", IEEE Trans. Circuits Syst., vol. 58, no. 5, pp. 947960, May 2011.

[4] Jiun-Ping Wang, Shiann-RongKuang, and Shish-Chang Liang, "High-Accuracy Fixed-Width Modied Booth Multipliers for Lossy Applications", IEEE Trans. Circuits Syst., vol. 19, no. 1, pp. 52-60, January 2011.

[5] Yuan-Ho Chen, T.-Y. Chang, and C.-Y. Li, "Area-Effective and Power-Efficient Fixed-Width Booth Multipliers Using Generalized Probabilistic Estimation Bias", IEEE Trans. Circuits Syst., vol. 1, no. 3, pp. 277-287, September 2011.

[6] Yuan-Ho Chen and T.-Y. Chang, "A High-Accuracy Adaptive Conditional-Probability Estimator for Fixed-Width Booth Multipliers", IEEE Trans. Circuits Syst., vol. 59, no. 3, pp. 594-603, March 2012.

[7] Shin-Kai Chen, Chih-Wei Liu, "Design and Implementation of High-Speed and Energy-Efcient Variable-Latency Speculating Booth Multiplier (VLSBM)", IEEE Trans. Circuits Syst. I, vol. 60, no. 10, pp. 26312643, October 2013.

[8] Shen-Fu Hsiao, Jun-Hong Zhang Jian, and Ming-Chih Chen, "Low-Cost FIR Filter Designs Based on Faithfully Rounded Truncated Multiple Constant Multiplication/Accumulation", IEEE Trans. Circuits Syst. II, Express Briefs, 2013.

[9] O. L. MacSorley, "High speed arithmetic in binary computers", Proc.IRE, vol.49,pp. 67-91, 1961.

[10] Parhami, Behrooz, "Computer Arithmetic: Algorithms and Hardware Designs", Oxford University Press 2000.

[11] David H. K. Hoe, Chris Martinez and Sri JyothsnaVundavalli, "Design and Characterization of Parallel Prefix Adders using FPGAs", Proc. IEEE, pp. 168172, 2011.

[12] Srinivasasamanoj R. ,M. Sri Hari and B. RatnaRaju, "High speed VLSI implementation of 256-bit Parallel Prefix Adders", International Journal of Wireless Communications and Networking Technologies, vol. 1, no. 1, 2012.

[13] Shelja Jose, ShereenaMytheen, "Modified Booth Multiplier Based Low-Cost FIR Filter Design", International Journal of Engineering Science and Innovative Technology, vol. 3, September, 2014.

[14] Phil E. Madrid, Brian Millar and Earl E. Swartzlander, Jr, "Modified Booth's algorithm for high radix fixed point multiplication", IEEE Transactions on Very Large Scale Integration (VLSI) Systems , Vol. 1, No. 2 June 1993.

[15] Soniya, Suresh Kumar, "A Review of Different Type of Multipliers and Multiplier-Accumulator Unit", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 4, July – August 2013

[16] GreeshmaHaridas, Dr. David Solomon George, "Area Efficient Low Power Modified Booth Multiplier for FIR", International Conference on Emerging Trends in Engineering, Science and Technology, Procedia Technology 24 (2016) 1163 – 1169