International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 1, January - 2014

Design Of Delay-Efficient Configurable Booth Multiplier For High Speed Applications

Shabeer Ahmad Ganiee
Assistant Professor, ECE
IUST, Awantipora, J&K

Sajad Ahmad Ganiee
M.Tech (Electronic Circuit and System)

Dr. Faroze Ahmad
HOD, Electronics & Communication
IUST, Awantipora, J&K

Abstract— ***Multipliers, play an important role in the design of microprocessor, graphical systems, multimedia systems, DSP system etc .Nearly 15 percent of total IC power is consumed by multiplication alone. It is therefore very important to have an efficient design in terms of performance, area, speed of the multiplier, and for the same Booth's multiplication algorithm provides a very fundamental platform for all the new advances made for high end multipliers meant for faster multiplication with higher performance. The algorithm provides an efficient encoding of the bits during the first steps of the multiplication process. In this paper an attempt has been made to design configurable logic for 4/8/12/16-bit booth multiplier. This multiplier can be configured to perform multiplication on 4 or 8 or 12 or 16 bit operands. The multiplier will detect the range of the operands through configuration register. The configuration register can be configured through input ports. The multiplier has been synthesized on Vertex 7 technology and it has achieved a maximum combinational delay of 1.846ns.***

Keywords — ***Booth Multiplier, Booth Multiplier Algorithm, Configurable Booth Multiplier (CBM)***

1. INTRODUCTION

Arithmetic and logic operations like addition, multiplication, exponensation play a vital role in digital circuits and have wide applications in the field of engineering. The demand for high speed processing has been increasing day by day as a result of expanding computer and signal processing applications. Higher throughput arithmetic operations are important to achieve the desired performance in many real-time signal and image processing applications [1]. Among these arithmetic operations, multiplication is the key of almost every digital circuit and finds applications in many Digital Signal Processing (DSP) systems such as Convolution, Fast Fourier Transform (FFT), filtering , in microprocessors in its arithmetic and logic unit and in graphics [2].

Digital multipliers are the most commonly used components in any digital circuit design. They are fast, flexible, reliable and efficient components that are utilized to implement any operation. Depending upon the arrangement of the components, there are different types of multipliers available. Particular multiplier architecture is chosen based on the application. Development of fast multiplier circuit has been a subject of interest over decades. Since multiplication dominates the execution time of most DSP algorithms, so there is a need of high speed multiplier[3,4].

Currently, execution time of multiplication is still the dominant factor in determining the instruction cycle time of a DSP chip. Many multiplication algorithms have been proposed in literature to perform multiplication, each offering different advantages and having tradeoff in terms of speed, circuit complexity, area and power consumption. Reducing the time delay and power consumption are very essential requirements for many applications [1, 5]**.**

Low power multipliers with high clock frequencies play an important role in today's digital signal processing [6,7,8]. That's why if one also aims to minimize power consumption, it is of great interest to reduce the delay by using various delay optimizations.

Various multiplication algorithms such as Booth, Array, Wallace tree, Braun and Baugh Wooley have been proposed in literature from time to time. Among these Donald Booth made an improvement in the multiplier by reducing the number of partial products generated. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed [9]. To speed up the multiplication Booth encoding performs several steps of multiplication at once. Booth's multiplication algorithm takes advantage of the fact that an adder, subtractor is nearly as fast and small as a simple adder.

In this paper, an attempt has been made to configure the Booth multiplier using configuration register that can supports single 4-bit, single 8-bit, single 12-bit or single 16-bit data. This CBM depends upon the output of configuration register that can be configured through input ports.Since there are sequential and combinational multiplier implementations but combinational case will be considered here because the scale of integration is large enough to consider parallel multiplier implementations in digital VLSI systems.

2. BOOTH MULTIPLIER

Andrew Donald Booth in 1951, devised a multiplication algorithm which was named after his name as Booth's Algorithm. Signed multiplication is a vigilant process. Through unsigned multiplication there is no need to take the sign of the number into consideration. Same procedure cannot be applied for signed multiplication due to the reason that the signed numbers are in a 2's compliment form which would

give us inaccurate result if multiplied in an analogous manner to unsigned multiplication [10].Unsigned multipliers cannot be applied to most of the multimedia and DSP applications due to their signed multiplication operation [11]. Thus here Booth's algorithm comes in rescue. The Booth recording multiplier scans the three bits at a time to reduce the number of partial products generated [12].Booth's algorithm conserves the sign of the end result, thus showing better performance in terms of operating speed ,time delay, power dissipation and area. From the basics of Booth Multiplication it can be proved that the addition/subtraction operation can be skipped if the successive bits in the multiplicand are same, thus reducing the delay to a greater extent.

2.1 Booth Multiplication Algorithm

Booth's algorithm multiplies two signed binary numbers in two's complement notation. Various steps involved in the Booth's multiplication are as:

Step 1: From the two numbers under test decide which operand will be multiplier and which will be the multiplicand .Note that the number with smallest difference between a series of consecutive numbers is chosen as a multiplier.

For example if we have to multiply 10 (01010) and -5 (11011). For 10 (01010) we have ----- 0 to 1 one change,1 to 0 one change,0 to 1 one change and 1 to 0 one change .So , in total there are four changes in binary form of 10.

For -5(11011),we have ---- 1 to 1 no change,1 to 0 one change ,0 to 1 one change and 1 to 1 no change ,so in overall there are only two change in -5.

Thus -5 (11011) is chosen as the **multiplier** and 10 (01010) as the **multiplicand.**

Step 2: Begin with the product that consists of the multiplier with an additional zero padding bits.Since our multiplier is 11011, after adding 5 leading zeros to the multiplier we get our beginning product.

00000 11011------ Beginning product

Step 3: Use least significant bit LSB and previous LSB to determine the arithmetic action.Intially 0 is chosen as the previous LSB.Thus our initial product and previous LSB becomes

00000 11011 0 ------ initial 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:

00 No arithmetic action

01 Add multiplicand to the left half of the product.

10 Subtract multiplicand from the left half of the product.

11 No arithmetic action

Place the result so obtained from arithmetic operations in the left half of the beginning product.

Step 4: Perform an arithmetic right shift (ASR) on the entire product for each pass. After X-passes we will get the required result, where X is the number of bits in the input operands. For the above example the result can be thus obtained after five passes.

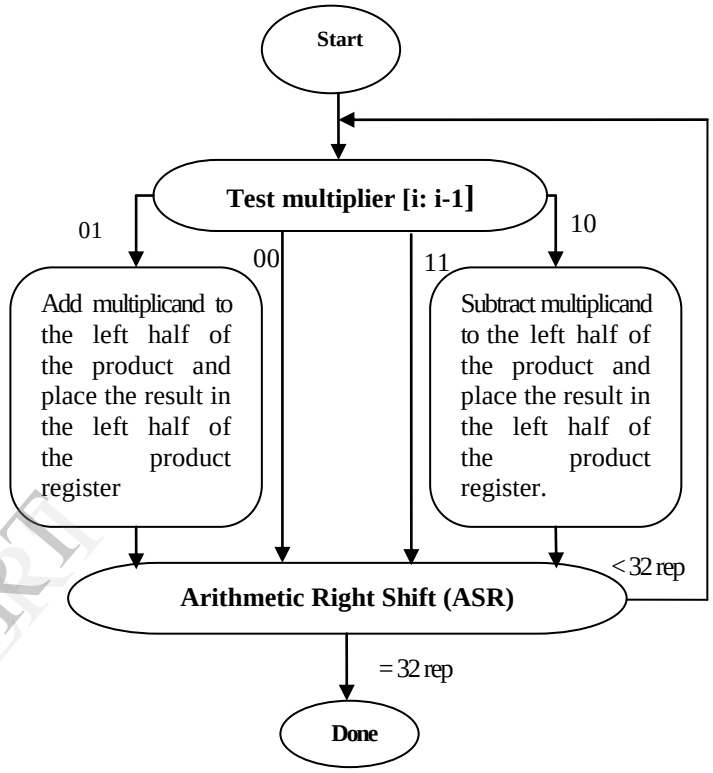

Figure 1 Flow Chart of Booth Multiplier

3. CONFIGURABLE BOOTH MULTIPLIER

3.1 Booth Multiplier Architecture

Booth multiplier architecture consists of various blocks each performing a certain task as:

1) A 16-bit register A that stores the value of multiplicand,16-bit parallel-load shift registers B that stores the multiplier initially, and the least significant 16 bits of the final multiplication product.

2) The 16-bit parallel-load shift registers ACCUMULATOR initially cleared is used for the storage of final product . ACCUMULATOR and B are concatenated so that the bit that are shifted out of ACCUMULATOR will be shifted into B. Also, the right shift operation is sign extended.For example, if ACCUMULATOR stores 1010101010101011 and B stores 0011111111000111,then

after concatenated right shift operation, ACCUMULATOR = 1101010101010101 and B =1001111111100011.

3) The 16-bit Airthmetic Logic unit (ALU) will perform the necessary arithmetic operations (addition ,subtraction etc) accoding to the control signal provided by CONTROL block after examining least two significant bits of the partial product and then pass its output to 16-bit ACCUMULATOR.

4) After examining least two significant bits of the partial products, the CONTROL block Y provides the necessary control actions.

5) Operation of Control block Y is controlled by 4-bit binary COUNTER that gives reference count to the CONTROL block.

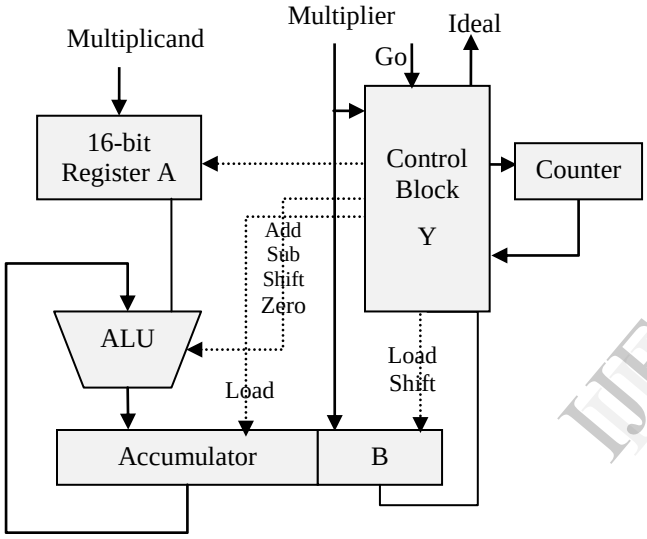

Figure 2 Booth Multiplier Architecture [13]

3.2 Configurable Booth Multiplier Working

From the basics of Booth multiplication algorithm we came to know that number of passes or cycles to obtain the final product depends upon the operand width. If the input data is of the form of 001110(multiplier) and 001001(multiplicand) we have to go for six passes to obtain the result .Since the significant data is contained in least 4 bits, output result can be obtained only after four passes by suppressing most significant bits, being zero .That will not only reduce the delay but also reduces the switching to a greater extent. So, an attempt has been made to make the Booth multiplier configurable to reduce the delay and power.

Initially, range of both the operands A and B are detected by Configuration register that is being configured through input ports. Configuration register will detect whether the computation will be done on 4 bit, 8 bit, 12 bit or 16 bit. After range detection, further computation will be done according to basic Booth's Algorithm. A register PA is taken, that will store the concatenation of accumulator (4 bit or 8bit or 12 bit or 16 bit, initially assigned with 0's), multiplier A (4 bit or 8 bit or 12 bit or 16 bit), and an extra least significant bit, LSB (initially assigned with 0).Least two significant bits of PA [1:0] are checked whether they are 00, 01, 10, and 11. If PA [1:0] = 01, then PA + B operation will occur with single ASR, and if PA [1:0] = 10, then PA – B will be calculated with single ASR, else if PA [1:0] =00 or PA [1:0] = 11 then no arithmetic operation will occur, only single arithmetic right shift will be done on PA. This whole process of checking of PA [1:0] bits will be done repeatedly and the number of iterations will depend upon the range detected. Finally, the final output will be saved in Accumulator.

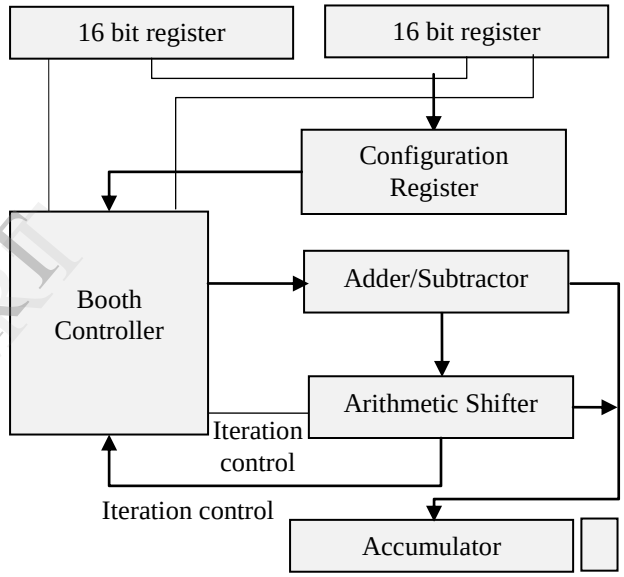

Figure 3 Working of Configurable Booth Multiplier

3.3 Range detection by Configuration Register

The configuration register will detects the effective dynamic range of input data and then generates the control signal to determine the flow of data. To simplify the implementation range detection can be realized by using the group of input bits. The data detection starts from the most significant bits, examining each four bit group. In the range detection technique, both the input 16-bit operands A [16:0] and B [16:0] are divided into four parts or subexpressions that are A [15:12], A [11:8], A [7:4] and A [3:0] and B [15:12], B [11:8], B [7:4] and B [3:0]. The size of both operands is checked separately and simultaneously whether they come in the range of 4 bits, 8 bits, 12 bits or 16 bits using the following relation:

If (| (A[15:12]) ==1) (i.e. Performing Bitwise OR operation on four MSB.)

```
        range A=2'b11
  else   if( |(A[11:8] ) ==1 )
     rangeA=2'b10;
     else if ( | (A[7:4] )==1)
       range A=2'b01;
 else if ( |(A[3:0] )==1)
       range A=2'b00;
     else if( |(A[11:8] )==1)
     range A=2'b10;
       else if( |(A[15:12] )==1)
         range A=2'b11
```

In the similar manner range is detected for B and the final range is decided by the relation:

```
if (range A > range B)
      range=range A
    else
     range= range B
```

This procedure of range detection minimizes the generation of partial products to a greater extent by suppressing the most significant bits if they are zero. Calculation will become shorter, faster and efficient..

4. RESULTS

4.1 Simulation Results

For the model under consideration the simulation results are carried out using Verilog HDL as simulation tool and Modelsim as simulator. Simulation results for various input operands A [15:0] and B [15:0] to obtain P [31:0] have been verified.

TABLE 1
.

RESULT FOR VARIOUS INPUTS OF DIFFERENT RANGE

A[15:0]	B[15:0]	P[31:0]	Range
1010 1000 0011 0011 (A_{dec}= -22477)	0011 0101 0100 0000 (B_{dec} = 13632)	11101101101111001 001101111000000 (P_{dec}= -306406464)	16
0000 0101 0100 0011 (A_{dec} = 1347)	0000 0001 1111 1110 (B_{dec} = 510)	00000000000010100 111101101111010 (P_{dec}= 686970)	12
0000 0000 0101 1111 (A_{dec}= 95)	0000 0000 0011 1010 (B_{dec} = 58)	00000000000000000 001010110000110 (P_{dec}= 5510)	8
0000 0000 0000 0110 (A_{dec}= 6)	0000 0000 0000 0111 (B_{dec}= 7)	00000000000000000 000000000101010 (P_{dec}= 42)	4

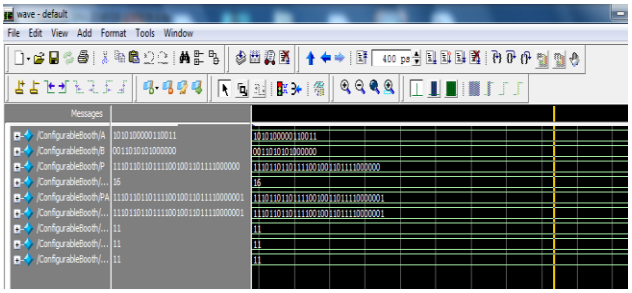

Figure 4 Simulation Result for A=1010 1000 0011 0011 and B= 0011 0101 0100 0000

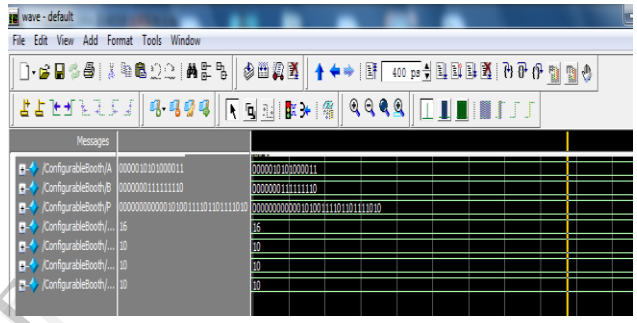

Figure 5 Simulation Result for A=0000 0101 0100 0011 and B= 0000 0001 1111 1110

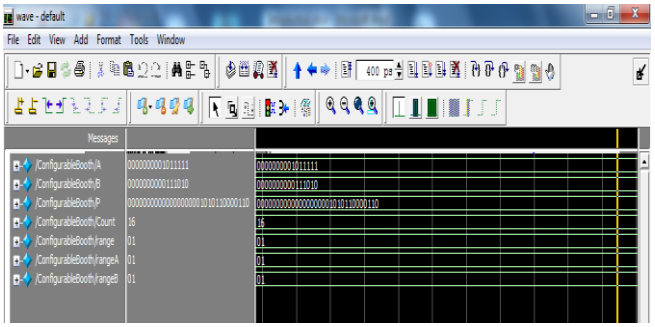

Figure 6 Simulation Result for A=0000 0000 0101 1111 and B=0000 0000 0011 1010

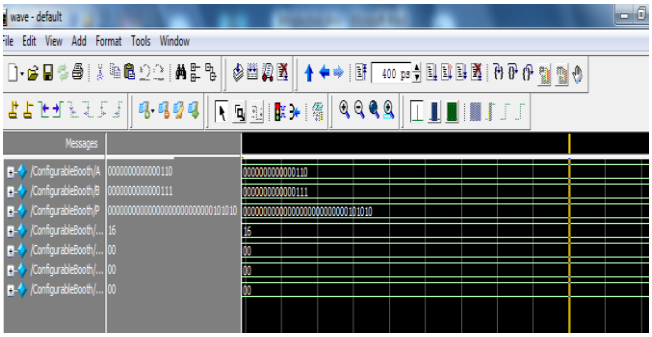

Figure 7 Simulation Result for A=0000 0000 0000 0110 and B=0000 0000 0000 0111

4.2 Synthesis

The multiplier has been synthesized on Vertex 7 FPGA Board. A detailed summary of devices utilized and timing summary has been shown below:

TABLE 2.

SUMMARY OF DEVICES UTILIZED IN CONFIGURABLE BOOTH MULTIPLIER

Parameter		Value
Latches	1-bit Latch	22
Comparators	2-bit Comparator	1
Multiplexers	1-bit 2-to-1 multiplexer	47
	2-bit 2-to-1multiplexer	13
	33-bit 3-to-1 multiplexer	1

TABLE 3.

TIMING SUMMARY

Delay	Value
Minimum input arrival time before clock:	1.042ns
Maximum output required time after clock:	1.575ns
Maximum combinational path delay:	1.846ns

From the above analysis of delay, Configurable booth multiplier is delay efficient .The Combinational delay of Configurable booth Multiplier is 1.84ns which is quiet low than the simple booth multiplier having a delay of 7-10ns.

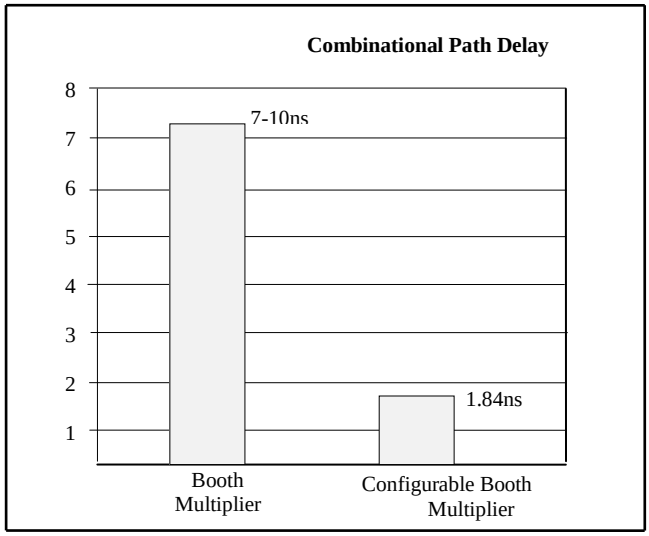

Figure 8: Graphical representation of Combinational Delays

5:CONCLUSION

We have presented a 4/8/12/16-bit configurable booth multiplier. This multiplier can be configured to perform 4 or 8 or 12 or 16 bit operands depending upon the output of configuration register. The multiplier will detect the range of the operands through configuration register. The configuration register can be configured through input ports. This process of configuration not only reduces the combinational path delay but also reduces power consumption to a larger extent. It also deactivates the redundant switching activities in ineffective ranges as much as possible. The proposed multiplier is very suitable for portable multimedia and DSP applications which require flexible processing ability, lesser switching activity and short design cycle. The multiplier has been synthesized on Vertex 7 FPGA Board and it has achieved a maximum combinational delay of 1.846ns.

REFERENCES

[1] Himanshu Thapliyal and Hamid R. Arabnia, "A Time-Area- Power Efficient Multiplier and Square Architecture Based On Ancient Indian Vedic Mathematics", Department of Computer Science, The University of Georgia, 415 Graduate Studies Research Center Athens, Georgia 30602-7404, U.S.A.

[2] Purushottam D. Chidgupkar and Mangesh T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing", Global J. of Engng. Educ., Vol.8, No.2 © 2004 UICEE Published in Australia.

[3] Low Power and High speed 8*8 bit Multiplier using non clocked Pass Transistor Logic,C.Senthilpari Ajay Kumar Singh and K Diwadkar 14244-1355-9/2007,IEEE

[4] Kiat-Sang Yeo and Kanshik Roy "Low voltage Low Power VLSI Sub System" Mc Graw- Hill Publication

[5] E. Abu-Shama, M. B. Maaz, M. A. Bayoumi, "A Fast and Low Power Multiplier Architecture", The Center for Advanced Computer Studies, The University of Southwestern Louisiana Lafayette, LA 70504.

[6] Padmanabhan Balasubramanian and Nikos E. Mastorakis, "High Speed Gate Level Synchronous Full Adder Designs," WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS Issue 2, Volume 8, 290-300, February 2009.
[7] Sanjiv Kumar Mangal,Rahul M.Badghare," FPGA Implementation of Low Power Parallel Multiplier",10th International Conference on VLSI Design,2007.
[8] Yingtao Jiang, Abdulkarim Al-Sheraidah, Yuke Wang, Edwin Sha, and Jin-Gyun Chung, " A Novel Multiplexer-Based Low-Power Full Adder," in IEEE transactions on circuits and systems vol. 51, no. 7, July 2004
[9] L.D. Van and C. C. Yang, "Generalized low-error area efficient fixed width multipliers," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.
[10] Laxman S, Darshan Prabhu R, Mahesh S Shetty ,Mrs. Manjula BM, Dr. Chirag Sharma,FPGA Implementation of Different Multiplier Architectures, International Journal of Emerging
[11] Shiann-Rong Kuang and Jiun-Ping Wang " Design of Power efficient Configurable Booth Multiplier Vol.57,No3,March1010
[12] Tam Anh Chu, "Booth Multiplier with Low Power High Performance Input Circuitary", US Patent, 6.393.454 B1,May 21, 2002.
[13] ECE/Comp. Sci. 352 – Digital System Fundamentals, Project #2 (Spring 2000), Department of Electrical and Computer Engineering, University of Wisconsin – Madison, April 2000