DESIGN AND IMPLEMENTATION OF A 32 BIT SIGNED BOOTH MULTIPLIER USING VERILOG HDL

PRIYANKA A. GOSWAMI
13BEC031
DEPT. OF ELECTRONICS AND COMMUNICATION
INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY.
AHMEDABAD, INDIA.

PRASHANT GANDHI
13BEC029
DEPT. OF ELECTRONICS AND COMMUNICATION
INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY.
AHMEDABAD, INDIA

ABSTRACT: **This paper presents an efficient design of a 32 bit signed Booth Multiplier and its implementation. The requirement for today's modern computers is high speed multiplier for signed and unsigned numbers. Thus, this paper presents the design and implementation of Booth multiplier which is suitable for high speed digital circuits. We also need to take care about the area and power consumed. Here we are using FPGA DE2 board for hardware implementation.**

KEYWORDS: **Booth Multiplier, Verilog, Signed bit, Quartus, Model Sim Altera.**

I. INTRODUCTION

Multiplication is the most commonly used operation in almost all computing systems. Infact multiplication is nothing but addition since, multiplicand adds multiplier no. of times to itself giving the multiplication value between multiplier and multiplicand. But such kind of operation requires a huge hardware and the operating speed is really low. Due to this power consumption is also more. As oppose to this, we require high speed and even less power consumption and minimum area. Among these three, speeds is the one which requires special consideration.

If we observe closely multiplication includes two steps one is producing partial products and another is adding of these partial products. To speed up the addition among the partial products we need faster adder architectures. The multipliers have a important role in the performance of entire system and hence, many high performance algorithms have been proposed. The high speed and dedicated multipliers are used in pipeline and vector computers. So many algorithms are there for multiplication such as Systolic algorithm, Robertston's algorithm, Booth algorithm, Wallace tree multiplier etc. But the Booth Algorithm is most commonly used because of the less delay time among the all others. The high speed and pipelined Booth multipliers are used in digital signal processing (DSP) applications such as multimedia and communication systems. High speed DSP applications such as FFT (Fast Fourier Transform) require addition and multiplication.

Booth's multiplication is an algorithm which multiplies two signed binary numbers in 2's compliment form. This algorithm was introduced by Andrew Booth in 1950 while he was researching on crystallography at Birkbeck College in Bloomsbury, London. He used desk calculators for fast add shifting and adding and created the algorithm with increased speed. Its main application is in computer architecture.

II. DIFFERENCE BETWEEN BOOTH MULTIPLIER AND OTHER MULTIPLIERS

As mentioned above today main concern is to reduce power consumption and area. One of the substitute to reduce power consumption is by the parallel processing is Systolic Array. Systolic array is a pipelining architecture based multiplier in order to reduce the power consumption. The Systolic Array is the network arrangement of the cells or units like the pipe processing. It gives the high speed as an benefit added with scalable. Even though it's so expensive and the process is too difficult to build up the multipliers.

Another method of multiplication is Wallace tree multiplication which is a tree based multiplier. When compare to the other multipliers Wallace tree multiplier is the high speed multiplier, it is the efficient method to the circuit operation. The latency is reduced by the carry save adder which is used in parallel multiplier as Wallace tree multiplier. Though it is the high speed it consumes more power than the booth multiplier.

III. THE ALGORITHM

Multiplication consists of three steps: 1.To generate the partial products; 2.To add the generated partial products until the last two rows are remained; 3.To compute the final multiplication results by adding the last two rows. Booth's algorithm examines adjacent pairs of bits of the 32-bit multiplier Y in signed 2's compliment representation, including an implicit bit below the least significant bit, Y+1 = 0. For each bit x[i] with i=0 to 31, the bits x[i] and Y+1 are considered. For these two are equal, the product accumulator P is left unchanged. When x[0] = 0 and Y+1 = 1, the multiplicand times 2^i is added to P; and when x[0] = 1 and Y+1=0, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product. The multiplicand and product are specified usually, and both are in 2's compliment form like the multiplier but any system supporting addition and subtraction may work well. There are many optimization and variations done to this algorithm. The algorithm is defined as converting 1's string of multipliers to higher order +1 or lower order -1 at the end of the string.

IV. RULES OF BOOTH ALGORITHM

Booth's algorithm is implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward arithmetic shift on *P*. Suppose M and N are the multiplicand and multiplier respectively, and X and Y representing the number of bits in M and N.

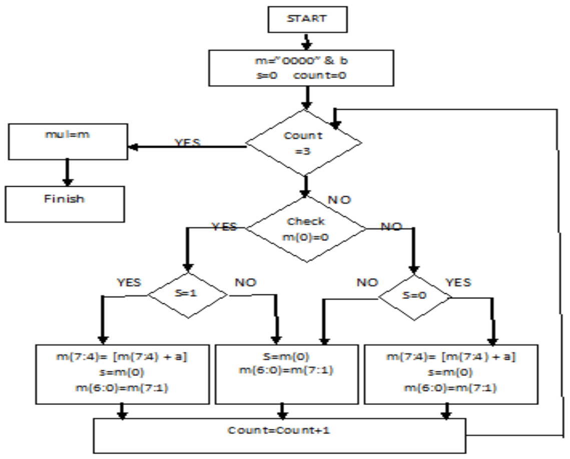
1) Determining the value of A and S along with initial value of P. all the values having a length of (X+Y+1).
    a. Fill the MSB of A with value of M and other Y+1 values with '0'.
    b. Fill the MSB of S with (-M) in 2's compliment form and fill other Y+1 bits with '0'.
    c. Fill the MSB X bits with '0' and to the right of it with the value of N and then fill the LSB with '0'.
2) Determine the two LSB's of P.
    a. If they are 01, find the value of *P+A*. Ignore any overflow.
    b. If they are 10, find the value of *P+S*. Ignore any overflow.
    c. If they are 00, do nothing. Use *P* directly in the nextstep.
    d. If they are 11, do nothing. Use *P* directly in the next step.

3) Arithmetically shift the value of P by a single bit to the right.

4) Repeat step 2) and 3) for Y times.

5) Drop the LSB bit from P and the value obtained is the product of M and N.

V. FLOW CHART



Here a=multiplicand
b= multiplier
m= product

VI. STATE DIAGRAM.

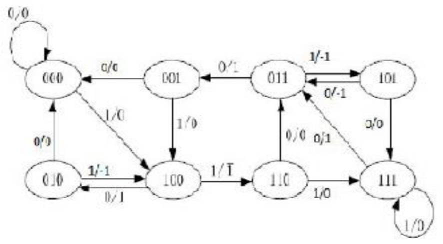


Figure 1. State diagram of 4-bit Booth Multiplier

VII. BOOTH MULTIPLIER CODE

```
module proj(a,b,c);
input signed[3:0] a;
input signed[3:0] b;
output signed [7:0] c;
reg signed[7:0] c;
reg [1:0] temp;
reg e;
reg [3:0] b1;
integer i;
always@(a,b)
begin
e=1'd0;
c=8'd0;
for(i=0; i<4; i=i+1)
begin
temp={a[i],e};
b1=-b;
case(temp)
2'd2:c[7:4]=c[7:4]+b1;
2'd1:c[7:4]=c[7:4]+b;
default:begin
end
endcase
c=c>>1;
c[7]=c[6];
e=a[i];
end
if(b==4'd8)
 c=-c;
 end
endmodule
```

VIII. CODE EXPLAINATION

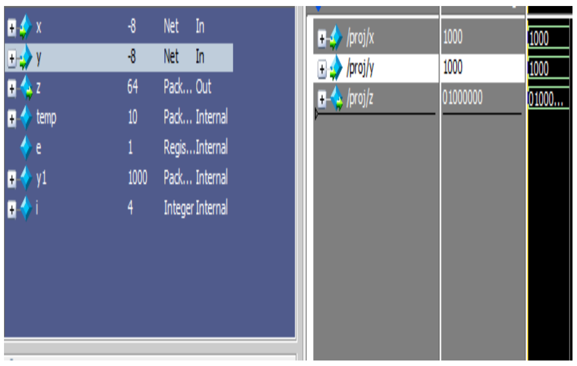
Here, this code is for 4-bit booth multiplier whose output will 8-bit. First we declare two inputs x and y where, x is multiplicand and y is multiplier. Here, inputs can be singed or unsinged number. Then we declare output z which is 8-bit output and it can be singed or unsigned number. Here, we are using behavioral level coding in Verilog. In always there are x and y which shows that if there is any change in x and y the code under always will be executed. Now, as we have discussed in algorithm we will first declare e which is Y+1 and z which is M with 0s. Now here because we are using behavioral level coding we will write statements between begin and end. Then we will use FOR loop to check the conditions. For this we will check Y+1 bit and x[0] bit and we will perform tasks mentioned in below table

Table 1. Task to be performed

x[0]	Y+1	Operation performed
0	0	Shift
0	1	Add y
1	0	Subtract y
1	1	Shift

Here after every addition and subtraction we will shift right. If 10 occurs we will first do 2’s compliment of y and then we will add it. The most important problem that will occur here is that whenever 1000 will occur it can be 8 or (–8) in such case we are putting one condition that whenever this will happen we will do 2’s compliment of the output. This code is for 4-bit similarly we can implement 32-bit booth multiplier also just by changing the input and output bit values.

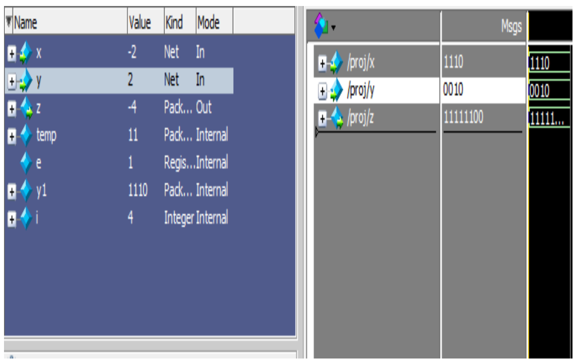
IX. SIMULATION RESULTS



```
x      -8     Net      In
y      -8     Net      In
z      64     Pack...  Out
temp   10     Pack...  Internal
e      1      Regis... Internal
y1     1000   Pack...  Internal
i      4      Integer  Internal
```

```
/proj/x   1000       1000
/proj/y   1000       1000
/proj/z   01000000   01000...
```

Figure 2. ModelSim RTL Simulation 1000x1000



```
Name   Value  Kind     Mode
x      -2     Net      In
y      2      Net      In
z      -4     Pack...  Out
temp   11     Pack...  Internal
e      1      Regis... Internal
y1     1110   Pack...  Internal
i      4      Integer  Internal
```

```
                     Msgs
/proj/x   1110       1110
/proj/y   0010       0010
/proj/z   11111100   11111...
```

Figure 3. ModelSim RTL simulation 1110x0010

Simulation results are as shown in the Figure 2. and figure 3. Figure 2. shows the multiplication of signed numbers. Here one bit (MSB) represents the sign of the number and others represent the magnitude of the number. One indicates that the number is negative and zero indicates that the number is positive. The inputs provided are 1110(-2) and 0010(2). The output obtained is 11111100(-4) is in 2‟s complement form. The sign bit should be made one for signed multiplication and it should be zero for unsigned multiplication Figure 3. shows the simulation result for signed numbers. The inputs given are 1000(-8) and 1000(-8) and the output obtained is 01000000(64). As per the our code if 1000 occurs which can be 8(decimal) or (-8)(2's compliment) in such
case we are taking 2,s compliment of the output.

X. RTL VIEW

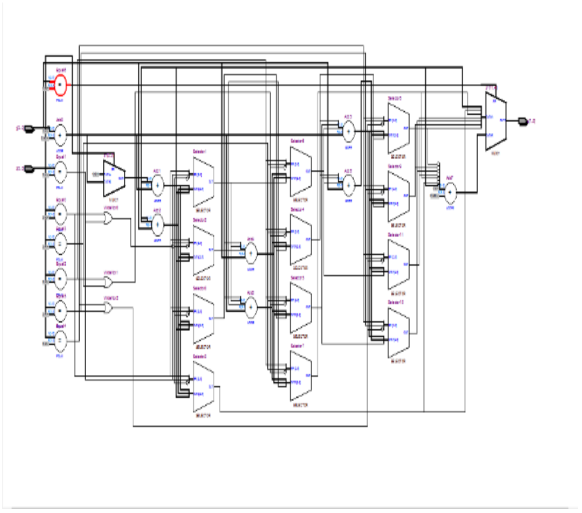


Figure 4. RTL view of 4-bit booth multiplier

From Figure 4. we can see that RTL view consist of eight adder blocks, fourteen multiplier blocks, seven equal blocks.

XI. FPGA is used for the design

The above stated results are accurate and error free. But, if it is time consuming then the efficiency of multiplier decreases because, in recent days the efficiency is measured not only with the accuracy but also with the speed. As FPGA is purely hardware circuit, the time taken by it to execute the algorithm is much less than the time used up by the softwares. Thus, the binary multiplication algorithm implemented on FPGA works faster than any other multiplication technology. Here, we are using EP2C35F672C6 device which is member of cyclone II family. Total 70 logical elements are created. Figure 5. is a FPGA DE2 board on which we can implement our booth multiplier code. From the figure we can see that it has 18 up-down switches and 18 LEDs. Now for the 4-bit booth multiplier first 4 switches will be assigned as multiplier which is x and next 4 switches will be assigned as multiplicand y. For the output z we will do Assignment of LEDs. Here input is 4x4 multiplication so output will come as 8-bit so 8 LEDs must have assigned here. Some of the outputs on the boards are shown in below figures.

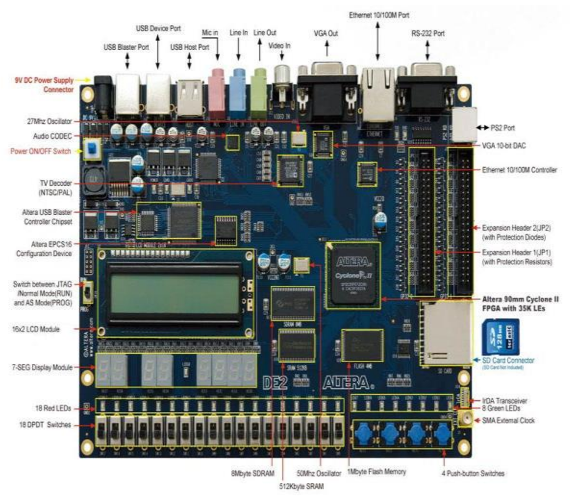


Figure 5. FPGA DE2 Borad

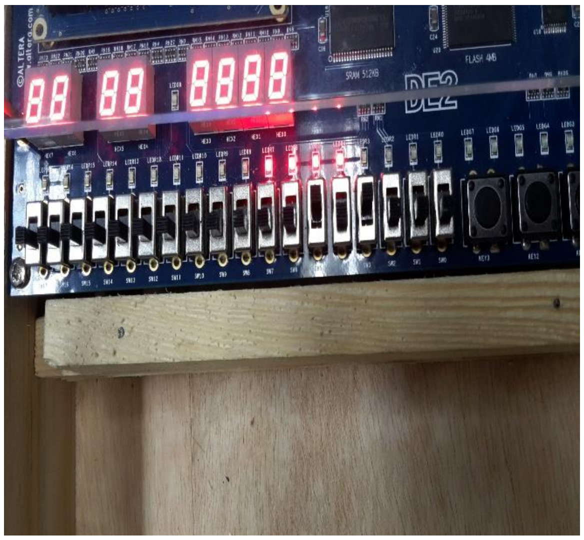


Figure 6. 1000x0010=11110000 on FGPA KIT

Here first four switches are assigned as 'a' which is multiplier and following to this switches next four switches are assigned as ‘b’ which is multiplicand and output is displayed as glow of LEDs.

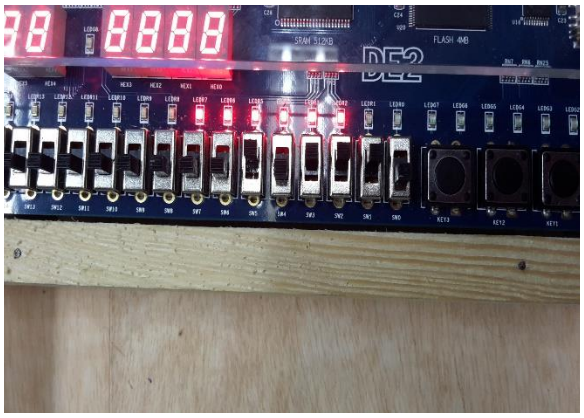


Figure 7. 1110x0010=11111100

XII. CONCLUSION

Booth multiplier is a multiplication algorithm to multiply two signed number and unsigned numbers. This paper proposes the model of a 32 bit Booth multiplier which is efficient and provides high speed and consumes less power. Booth multiplier use less area and also provide less delay than the others multiplication techniques.

XIII. REFERENCES

[1] Kavita, Jasbir Kaur “Design and Implementation of an Efficient Modified Booth Multiplier using VHDL” Proceedings of 2nd International Conference on Emerging Trends in Engineering and Management, ICETEM 2013.

[2] Dr. Ravi Shankar Mishra, Prof. Puran Gour, Braj Bihari Soni “Design and Implements of Booth and Robertson’s multipliers algorithm on FPGA” International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 1, Issue 3, pp.905-910

[3] Jayashree Taralabenchi, Kavana Hedge, Saumya Hedge “implimantation of binary multiplication using booth and systolic algorithm on FPGA using VHDL” International Conference & Workshop on Recent Trends in Technology, (TCET) 2012.

[4] academic.regis.edu/psmallwo/sitepages/cs440/.../wk1online-boothsalg.pptx.

[5] www.dauniv.ac.in/downloads/.../CompArchCh03L05BoothAlgor.pdf.

[6] enhanceedu.iiit.ac.in/wiki/images/Booth's_multiplier_-_task1_hint.pdf

[7] Analysis of Low Power, Area and High Speed Multipliers for DSP Applications K. S. Ganesh Kumar1, J. Deva Prasannam2, M. Anitha Christy3 1,2,3Department of Electronics and Communication & Karunya University, Coimbatore, T.N, India.