International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3 Issue No.5, pp : 561-563 1 May 2014

Design and Implementation of Booth Multiplier and Its Application Using VHDL

Akanksha Sharma, Akriti Srivastava, Anchal Agarwal, Divya Rana ,Sonali Bansal

SRMS Women's College of Engineering and Technology, Bareilly
Email:akanksha.sharma15feb@gmail.com[1]akriti2131@gmail.com[2]anchalagarwal92@gmail.com[3]
divyarana.sr@gmail.com[4]sonalibansal23@gmail.com[5]

ABSTRACT

Low power consumption and small area are some of the most important criteria for design of any high performance systems[i]. So in this paper the best solution to the problem is determined by designing a high speed multiplier chiefly booth multiplier which reduces the number of flip flops and memory size in the design circuitry as compared to conventional serial multiplier. Then implementation of a calculator using booth multiplier and several other operational modules is done using codes written in VHDL language using ISE XILINX 6.1 and simulated in MODEL SIM 5.4a.

Keywords:

booth multiplier, calculator, Xilinx, modelsim , low power consumption multipier, Reduced area multiplier, serial multiplier, high speed multiplier.

I. Introduction

Multipliers are the main component of many high performance systems such as calculators, digital signal processing applications, filters, microprocessors, etc. Performance of any system is generally determined by the performance of the multiplier used in it because the multiplier is generally the slowest element in any system.

With the addition of a number with itself by a given number of times, process of multiplication is done[ii]. The main steps of multiplication process are:

- Partial product generation
- Partial product reduction
- Final addition

A multiplication process following the above three steps results in more delay and large memory used.

As an alternative of this type of multiplication, there is a another multiplier booth multiplier based on the Booth algorithm given by "Andrew Donald Booth" as discussed below in Fig 1 for unsigned numbers and Fig 2 for signed numbers to reduce the number of partial products generated by taking into account two bits of the multiplier at a time, results in speed advantage over other multiplier architectures such as serial multiplier.

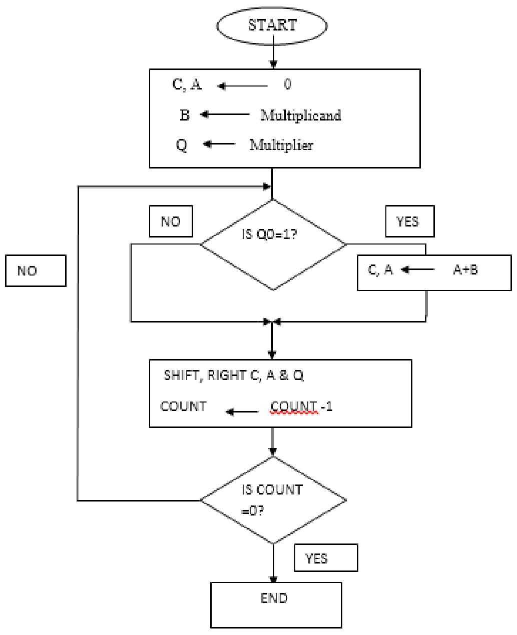


Fig1: Flowchart for booth's algorithm of unsigned number

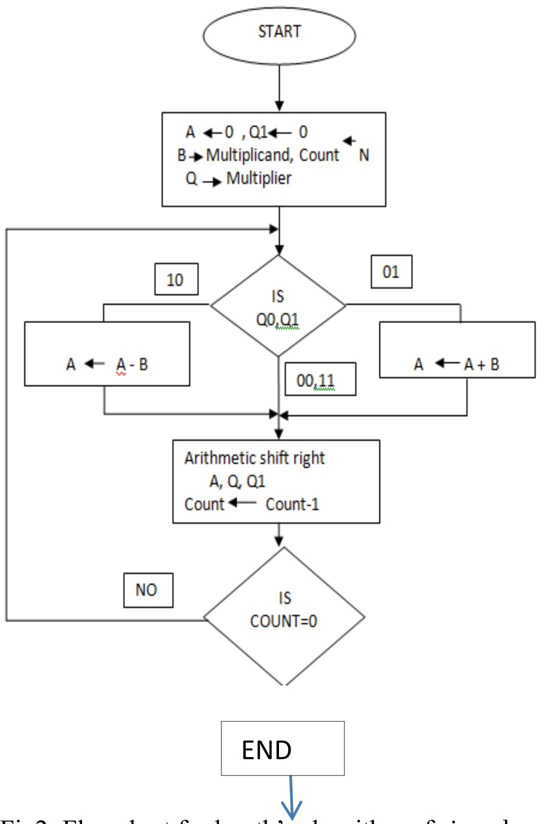


Fig2: Flowchart for booth's algorithm of signed number

Booth multiplier works procedurely as follows:

1. If number of bits of multiplicand is *u*, and that of multiplier is *v*.

2. Firstly, draw a three row grid each with columns for *u*+*v*+1 bits. The name is given as A (add), S (subtract), and P (product).

3. The first *u* bits of each row is filled i.e. for row A: the multiplicand, for row S: the 2's compliment of multiplicand and for row P: zeroes.

4. The next *v* bits of each row is given with:

- A: zeroes
- S: zeroes
- P: the multiplier

5. The last bit of each row is given with a zero.

Example: Find 2 × 4 with *u* = 4 and *v* = 4:

A = 0010 0000 0

S = 1110 0000 0

P = 0000 0100 0

Execution these steps *v* times, and do accordingly as the last two bits of product P are obtained...

00 or 11: only arithmetic shifting.

01: P = P + A. Ignore any overflow and shift it arithmetically.

10: P = P + S. Ignore any overflow and shift it arithmetically.

LSB is ignored rest bits are the final product of multiplication of two signed numbers.

Synthesis and Implementation

Steps to be performed on Xilinx for code synthesis[v] and implementation are as follows-

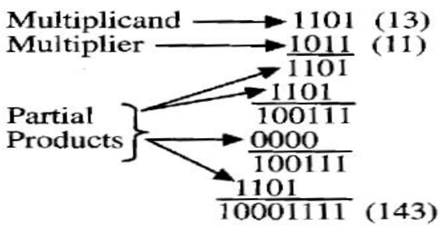
1. Entering VHDL code

2. Synthesis and implementation

3. Creating test benches

4. Simulation with modelsim

When the product is of form A*B, the first operand (A) is called as multiplicand, and second operand (B) is called as multiplier. Binary multiplication requires only shifting and addition. Example of binary multiplication:-



Multiplication of two 4-bit numbers requires

- a 4-bit multiplicand register
- a 4-bit multiplier register
- a 4-bit full adder
- an 8-bit register for product.

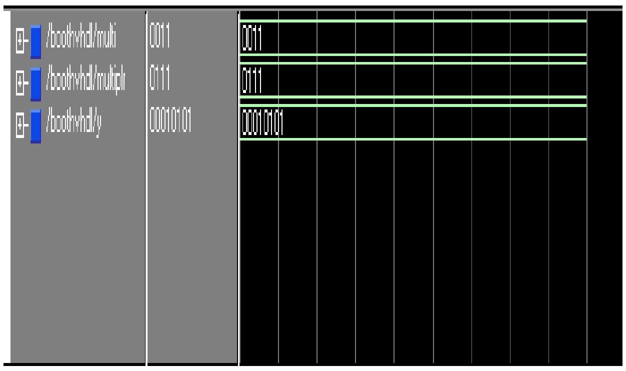
Booth multiplier performs multiplication by checking the last 2 bit of multiplicand to satisfy various conditions and repeat the same by 4 times.
The product register serves as an accumulator to accumulate the sum of partial products. Results simulated in modelsim are shown in Fig 3.
Then for the designing of calculator, all other module of operation are implemented this booth multiplier is implemented in a calculator using several other operational modules such as 4 bit parallel addition, parallel subtractor, division, squaring, cubing and other logical functions such as AND,OR,NOT etc. Another calculator using serial multiplier is also designed using same operational modules and a comparison has been made on the basis of number of slices, look up tables(LUTs) etc.

These operational modules are combined into a single package where different functions of these modules are created and then these functions are called into a VHDL module using different values of a select line. Select line's' has been assigned value from 0 to 15 and each module is operational with a definite value.

Example: for s = 0000 booth multiplier is operated, s= 0001 parallel addition is performed and so on.

A test bench is generated using test bench module then these test benches are simulated in modelsim. Below figure shows the simulated waveforms.

II. Results and tables



```
/boothvhdl/multi     0011       0011
/boothvhdl/multiplr  0111       0111
/boothvhdl/y         00010101   00010101
```

Fig 3: Simulation of booth multiplier

Fig 3 shows the simulation result of booth multiplier for two 4 bit inputs multi(3) and multiplr(3) and 8 bit output are store in signal y(9).

A test bench waveform is one which shows the result of specified input after a defined delay. Fig 4 shows the simulation result of calculator using booth multiplier and Fig5 shows the simulation result of calculator using serial multiplier. Both the simulated result is of test bench waveform to perform all the operation of calculator after a specific delay of 100ns.

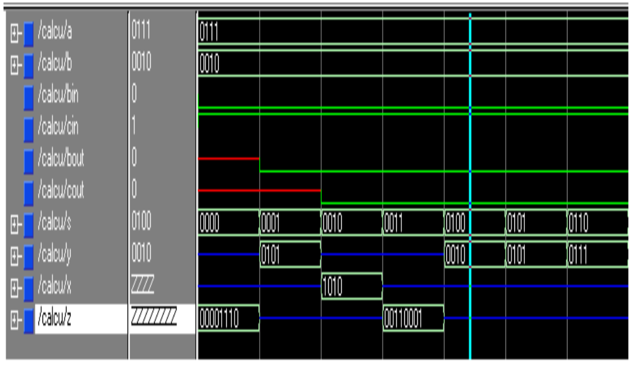


Fig4: Simulation result of calculator using booth multiplier

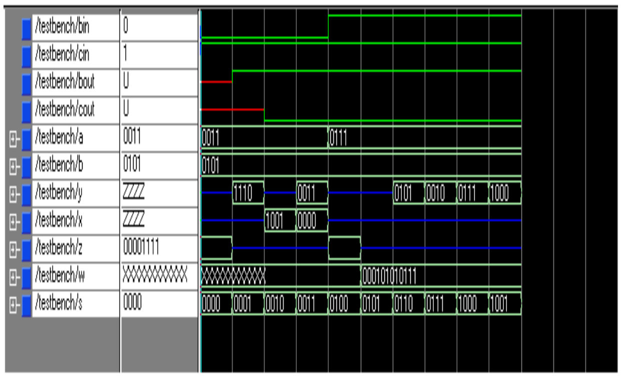


Fig5: Simulation result of calculator using serial multiplier

Here, in the calculator simulation window there are two inputs ‘a’(3) and ‘b’(5), and a select line ‘s’ to select the operation to be performed and corresponding result are store in the variable ‘x’, ‘y’and ‘z’.

- ‘x’(9) store the result of addition for ‘s’(0010) and cin(1).
- ‘y’(-2)store the result of subtraction for ‘s’(0001).
- ‘z’(15)store the result of multiplication for ‘s’(0000).
- in case of division ‘x’(1)& ‘y’(2) store the quotient and remainder for ‘s’(0011).

The result of logical operation is store by ‘y’.

As a result, both the calculators are synthesized and a design summary for both the multipliers are obtained which shows the number of LUTs and slices are used by that type of multiplication.

Logic utilisation	Used	available	utilisation
No of slices	24	960	2%
LUTsS	43	1920	2%
Number of bonded IOB	16	108	14%
memory	62436 kb	-	-

Table 1 : Design summary of Booth multiplier

As the utilization factor of booth multiplier is higher than that of serial multiplier but the memory utilization of booth multiplier is less than that of serial multiplier.

Logic utilisation	used	available	utilisation
No. of slices	12	960	1%
LUTs	23	1920	1%
No. of bonded IOB	16	108	14%
memory	62500 kb	-	-

Table 2: Design summary of Serial multiplier.

III. Conclusion

Multipl-ier type	Speed	Comple-xity	Layout	Area	Mem-ory
Serial	Low	Simple	Regular	Large	More
Booth	High	Comple-x	Irregul-ar	Medi-um	Less

Table 3 :Comparison between two multipliers on basis of different factors

Hence booth multiplier is satisfying its criteria of design for reduced area,power consumption and memory.

Multiplier	Multiplexers(2:1)	Adders (4-bit)
Serial	9	3
Booth	1	7

Table 4 : Mux and Adders used in each multiplier

It is seen from the Table 4 that no of multiplexers and adders utilized in booth multiplier is less as compared to the serial multiplier. So the objective of designing this multiplier is fulfilled as in large circuit design use of booth multiplier in place of serial multiplier considerably reduces the number of elements utilized in its design. Memory utilization of booth multiplier is less than that of serial multiplier i.e. for the case if booth reduces the memory usage by 2 % in a design of single gate so the reduction would be about of 10% in a large circuit design comprising of number of gates.[x]

IV. References

i. *Weste, Neil H.E. Eshraghian, and Kamran, “CMOS VLSI Design.*

ii. *ParhamiBehrooz . Computer arithmetic algorithm and hardware design.*

iii. *Circuit design with VHDL byVolneiA.Pedroni.*

iv. *Verilog Digital system Design 2e ZainalabedinNavabi*

v. *VHDL Programming By Examples Douglas Perry*

vi. *K. Babuluet al, G. Parasuram "FPGA Realization of Radix-4 Booth Multiplication Algorithm for High Speed Arithmetic Logics", (IJCSIT) Vol. 2 (5) , 2011, 2102-2107*

vii. *A. R. Cooper, “Parallel architecture modified Booth multiplier”,Proc.Inst.Electr.Eng.G, vol.135,*

a. *pp.125–128, 1988*

viii. *Wikipedia.*

ix. *A. D. Booth “ Asingned binary multiplication technique”.*

x. *Roy Kaushik,Yeo and Kiat-Seng “ Low Power vlsi Sub systems”.*