Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 6, June -2017

Implementation of Booth's Algorithm on FPGA

Mr. ShubhamMandhare
AISSMS(IOIT)

Mr. AdityaParadhe
AISSMS(IOIT)

Mr.PratikNiwalkar
AISSMS(IOIT)

Abstract - *The Booth's algorithm generates a 2n-Bit product and treats both positive and negative 2's complement n-bit operands uniformly. Booths Algorithm is used for the purpose of Binary multiplication. DSP processors can be used for implementing Booth's algorithm but since they follow sequential execution of instructions are slow in operation. FPGAs, on the other hand follow parallel execution of statements, which make them faster in operation.DSP processors are not designed to be AREA and POWER efficient, while FPGAs offer the well-known VLSI Design metrics of SPEED, AREA & POWER at low cost, low power even while handling high computational workloads.*

Thus we chose FPGA over DSP Processor for implementing the Booth's algorithm.

Keywords *- Booth's Algorithm, Multiplication, Field Programmable Gate Array, Ripple Carry adder, Carry look Ahead Adder, Structural Modeling Style, Behavioral Modeling Style.*

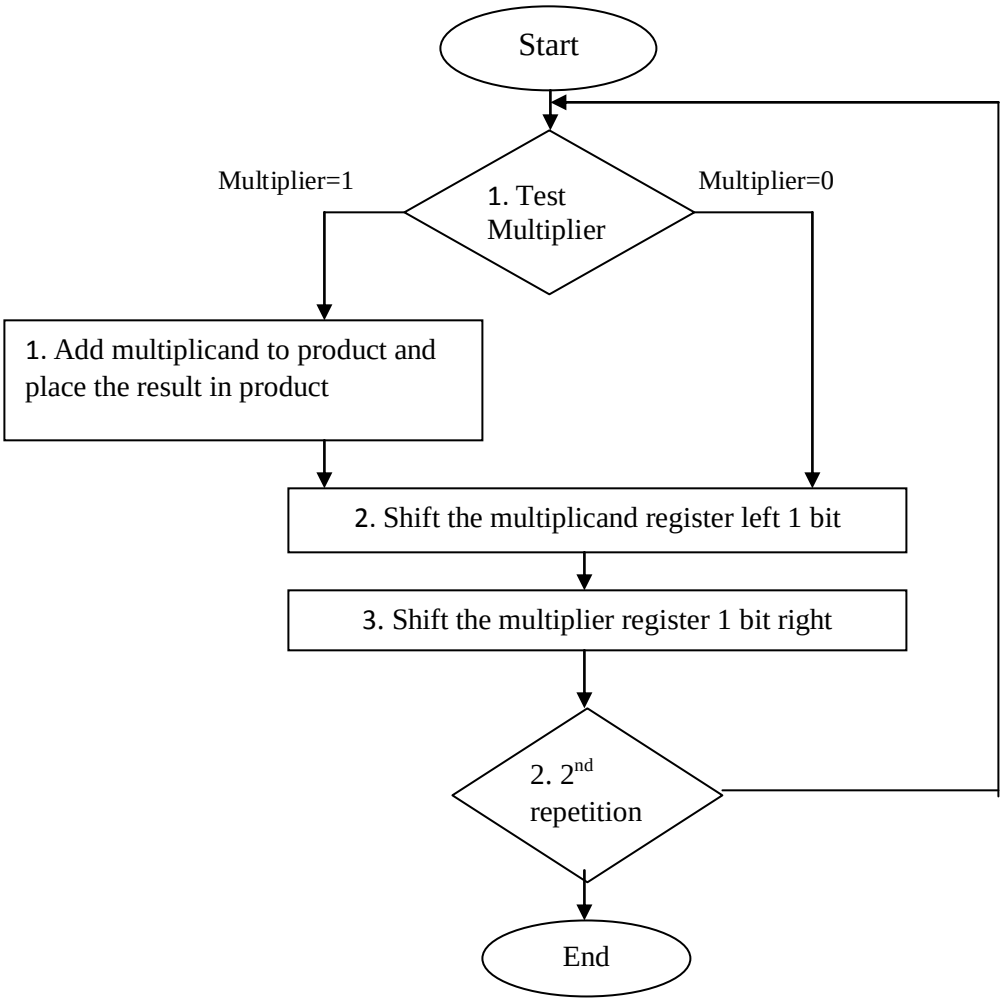
I. INTRODUCTION

In digital computing systems multiplication is an essential arithmetic operation. The multiplication operation consists of simply producing partial products and then adding these partial products the final product is obtained. Thus the speed of the multiplier depends on the number of partial product and the speed of the adder. As the multipliers are having a significant impact on the performance of the entire system, many high performance algorithms and architectures have been proposed. Hence multiplier is an important element of the digital signal processing such as convolution and filtering operations. The high speed Booth multipliers and pipelined Booth multipliers are used for digital signal processing (DSP) applications such as for multimedia and communication systems. We are using FPGA Because FPGAs offers high performance and very high operating speeds with limited amount of logic devices and IP cores available on the system. As FPGA is purely hardware circuit, the time taken by it to execute the algorithm is much less than the time taken by the software. Thus, the binary multiplication algorithm implemented on FPGA works faster than any other multiplication technology.Booth's algorithm is a powerful algorithm for signed number multiplication, which treats both positive and negative numbers uniformly. Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation.

II. BASIC BINARY MULTIPLICATION

Multiplier circuits are found in every computer, cellular telephone, and digital audio/video equipment. In fact, essentially any digital device used to handle speech, stereo, image, graphics, and multimedia content contains one or more multiplier circuits. The multiplier circuits are usually integrated within microprocessor, media coprocessor,
and digital signal processor chips. These multipliers are used to perform a wide range of functions such as address generation, Discrete Cosine Transformations (DCT), Fast Fourier Transforms (FFT), multiply-accumulate, etc. As such, multipliers play a critical role in processing audio, graphics, video, and multimedia data.

A multiplying circuit is able to perform a multiplication of n-bits X n-bits at a high speed by increasing the speed of the forming process of the partial products so that the delay time may be inhibited from increasing for a large n, and which can inhibit the chip size becoming large.Multiplication is more complicated than addition, being implemented by shifting as well as addition. If the number of partial products generated during multiplication are more in number, the system requires more time and more circuit area to compute, allocate, and sum the partial products to obtain the multiplication result. Fig.1 shows the flow chart for basic binary multiplier.



Flowchart No.1

With the recent advancements, the method of multiplication is divided into two basic steps-create a group of partial products, then add them up to produce the final product. Different ways of adding the partial products were mentioned, but little was said about how to generate the partial products to be summed. A recoding scheme introduced by Booth reduces the number of partial products by about a factor of 2.

ADDERS FOR MULTIPLICATION:

Fast carry propagate adders are important to high performance multiplier design in two ways. First, an efficient and fast adder is needed to make any "hard" multiples that are needed in partial product generation. Second, after the partial products have been summed in a redundant form, a carry propagate adder is needed to produce the final non redundant product.

A. Ripple Adder:

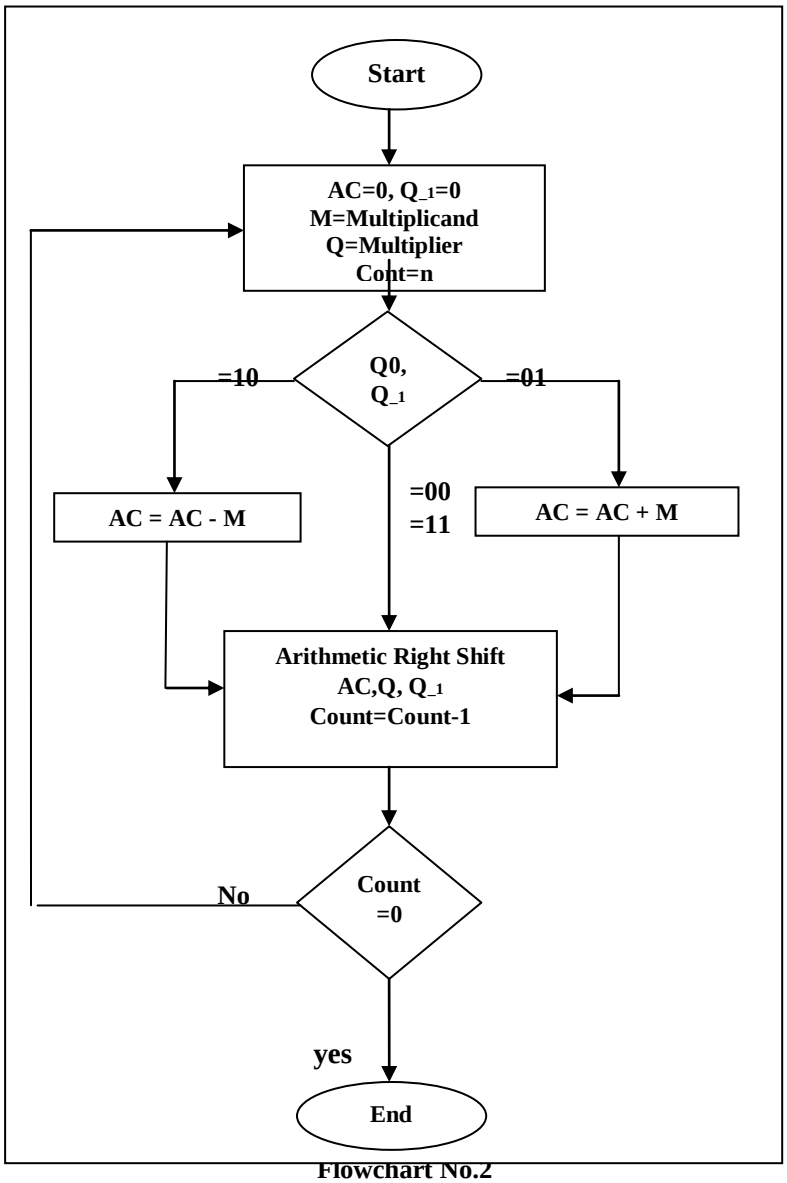
N bit numbers are added by designing a circuit using multiple Full adders. Each full adder inputs a Cin which is the Cout of the previous adder. This kind of adder is a ripple carry adder, since each carry bit "ripples" to the next full adder. Note that the first (and only the first) full adder may be replaced by a half adder in some cases. The layout of a ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic.

B.Carry Look-Ahead Adder (CLA):

The concept behind the CLA is to avoid therippling carry present in a conventional adderdesign. The rippling of carry produces unnecessarydelay in the circuit. Carry look-ahead logic uses theconcepts of generating and propagating carries.Although in the context of a carry look aheadadder, it is most natural to think of generating andpropagating in the context of binary addition, theconcepts can be used more generally than this. Inthe descriptions below, the word digitcanbereplaced by bit when referring to binary addition.

A circuit that multiplies two unsigned n bit binary numbers uses a 2 dimensional array of identical subcircuits. Each of which contains a full adder and an "and" gate. For large number of bits this approach may not be appropriate because of the large number of gates needed. Another approach is to use shift register in combination with an adder to implement the traditional method of multiplication

III. BOOTH'S ALGORITHM



Flowchart No.2

Signed multiplication is a vigilant process. Through unsigned multiplication there is no need to take the sign of the number into consideration. Even though in signed multiplication the same procedure cannot be applied for the reason that the signed number is in a 2's compliment form which would give in an inaccurate result if multiplied in an analogous manner to unsigned multiplication.

Thus here Booth's algorithm comes in. Booth's algorithm conserves the sign of the end result. While doing multiplication, strings of 0s in the multiplier call for only shifting. While doing multiplication, strings of 1s in the multiplier need an operation only at each end. We require to add or subtract merely at positions in the multiplier where there is a switch from 0 to 1 or from 1 to 0. In the following flow chart we have, b=Multiplier, a=Multiplicand, m= Product.

Now here we will require twice as many bits in our product as we already have in our two operands. The leftmost bit of our operands of both the multiplicand and the multiplier is always a sign bit, and can't be used as part of the value. Then choose which operand will be multiplier and which will be multiplicand. If one operand and both are negative then they are represented in two's complement form. Start in on with a product that consists of the multiplier in the company of an additional X leading zero bits. Now check the LSB and the previous LSB of product to find out the arithmetic action. Add '0' as the previous LSB if it is the FIRST pass.

Probable arithmetic actions are if:

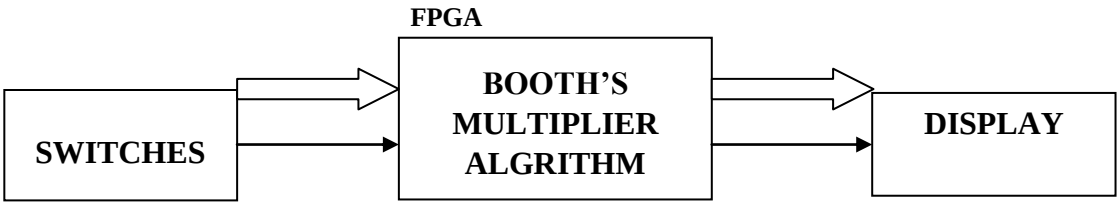
00:- no arithmetic operation is performed only shifting is done.

01:- add multiplicand to left half part of product and then shifting is done.

10:- subtract multiplicand from left half part of product and then shifting is performed

11:- no arithmetic operation is performed only shifting is done.

IV. PROPOSED ARCHITECTURE



V. SIMULATION RESULT

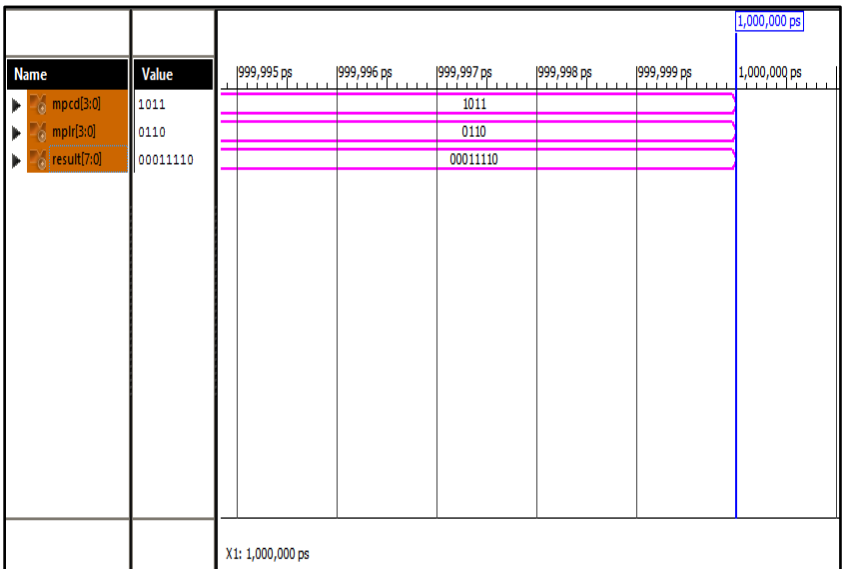


Fig.2.Simulation Result of 4 Bit Multiplier

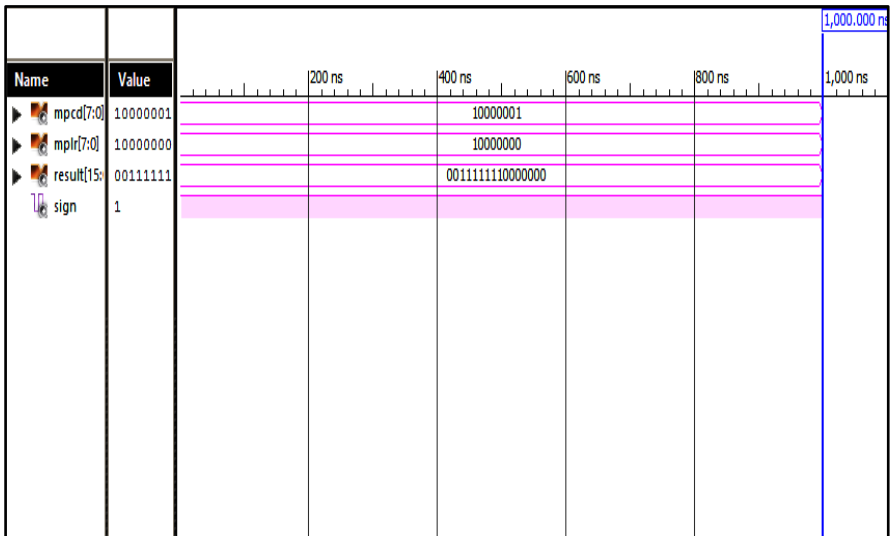


Fig.3.Simulation Result of 8 Bit Multiplier

VI. RTL SCHEMATIC

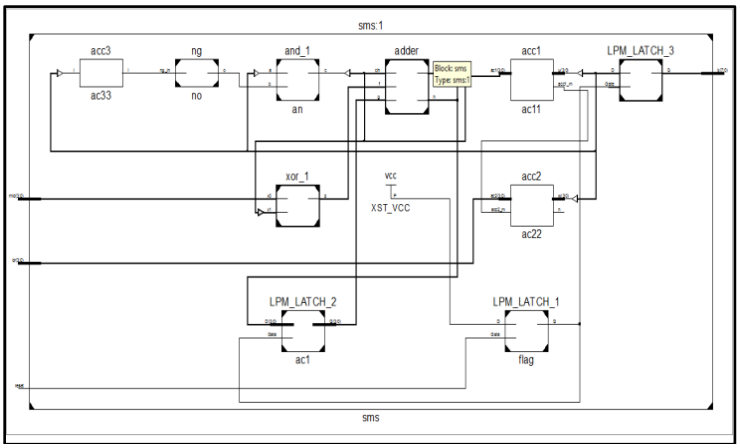


Fig.4.4 Bit Booths Multiplier Using Structural Modeling Style.

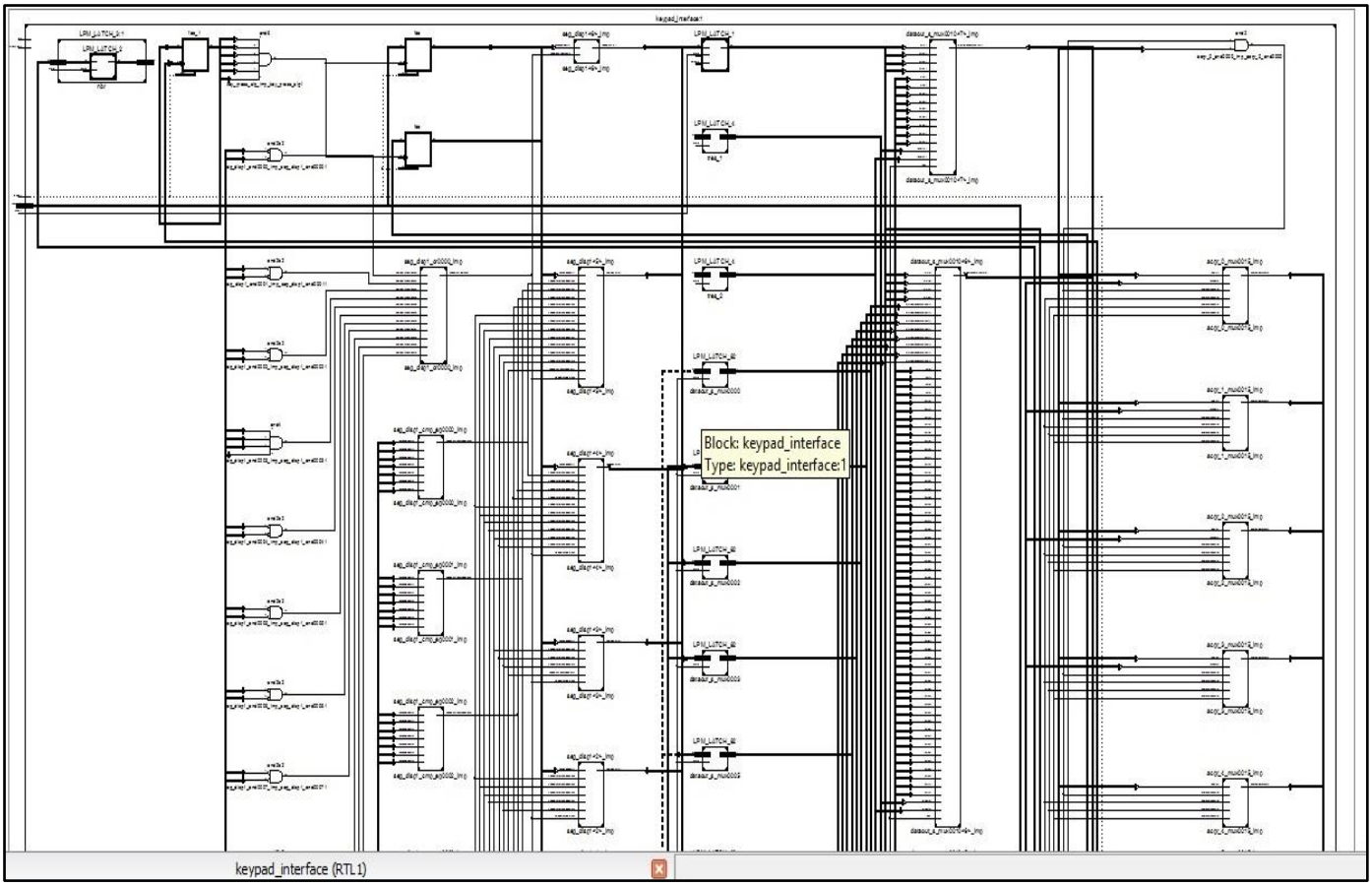


Fig.5. 4 bit Booths multiplier using behavioral modeling style.

VII. TECHNOLOGY SCHEMATICS

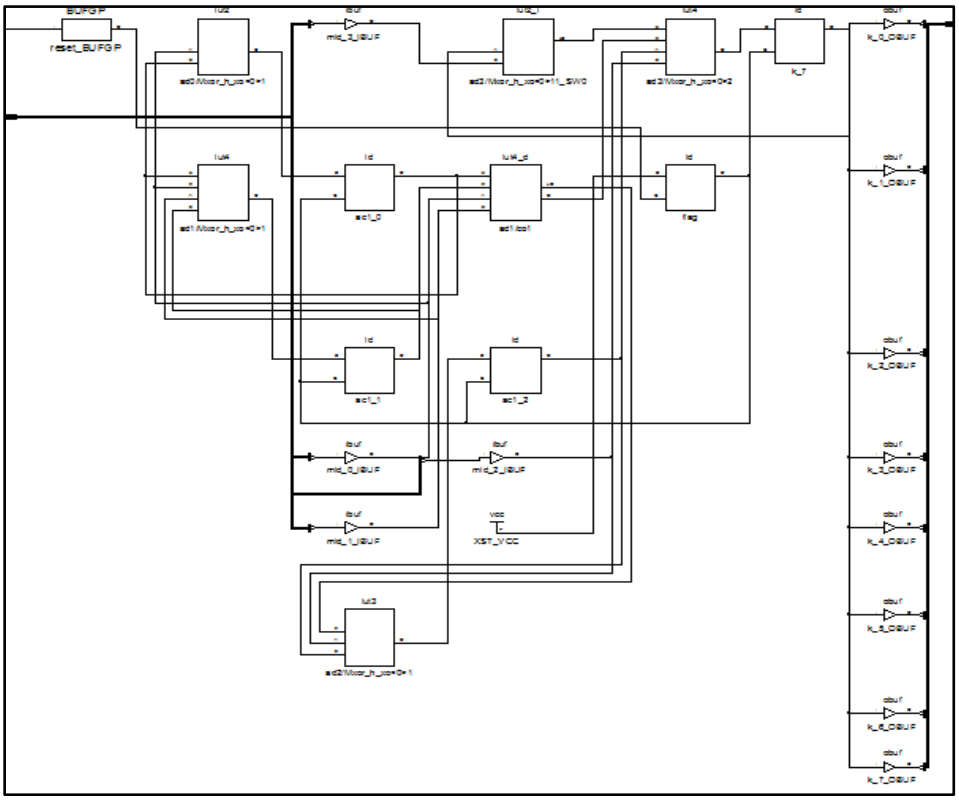


Fig.6.4 Bit Booths Multiplier Using Structural Modeling Style.

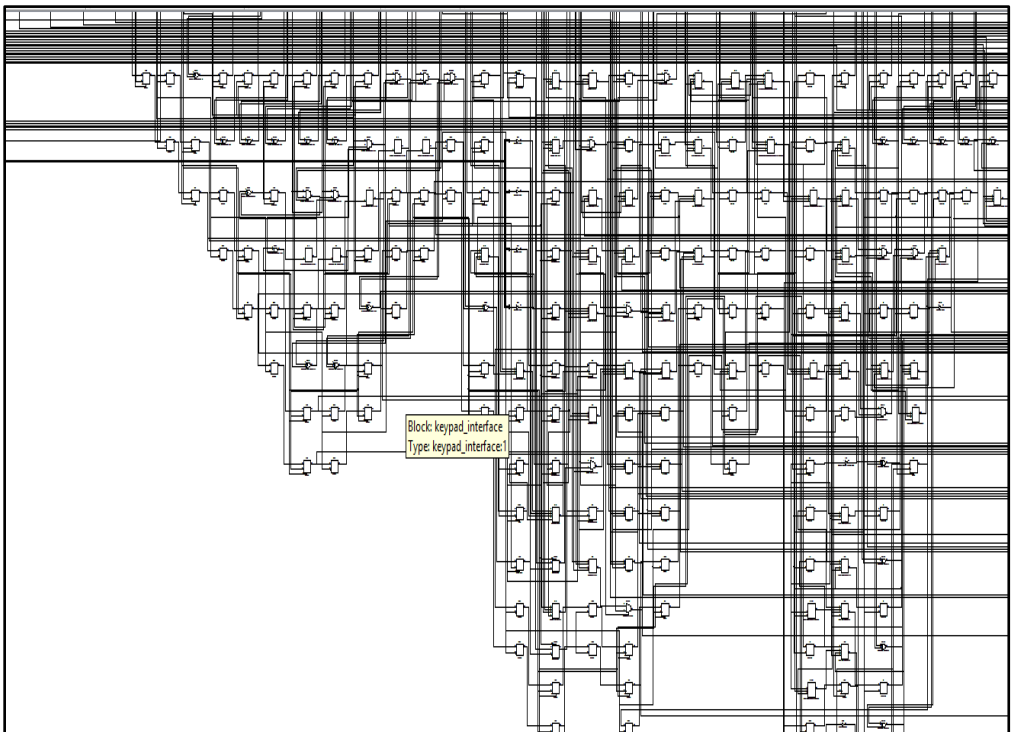


Fig..7. 4 Bit Booths Multiplier Using Behavioral Modeling Style (Using Keyboard-LCD)

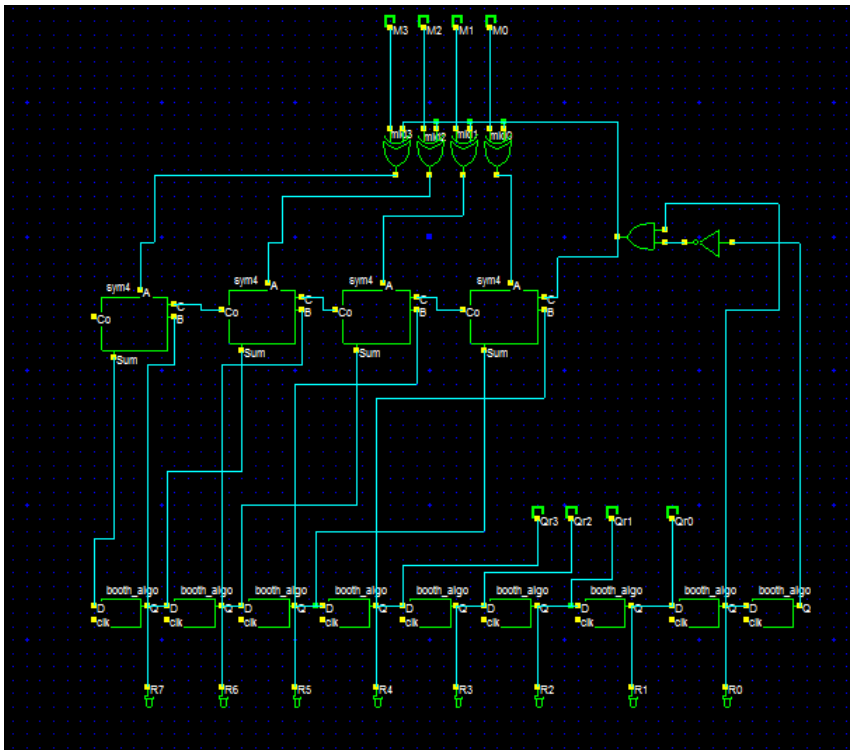


Fig.8.DSCH3 Layout for 4Bit multiplier

VIII. DEVICE UTILIZATION SUMMARY

Logic Utilization	Used	Utilization
Number of 4 input LUTs	44	1%
Number of occupied Slices	23	1%
Number of Slices containing only related logic	23	100%
Number of Slices containing unrelated logic	0	0%
Total Number of 4 input LUTs	45	1%
Number used as logic	44	
Number used as a route-thru	1	
Number of bonded IOBs	17	10%
Average Fan out of Non-Clock Nets	4.00	

Table.1 4 Bit Booth's Multiplier (Without Keyboard and LCD)

Logic Utilization	Used	Utilization
Total Number Slice Registers	256	5%
Number of 4 input LUTs	192	3%
Number of occupied Slices	215	8%
Number of Slices containing only related logic	215	100%

Number of Slices containing unrelated logic	0	0%
Total Number of 4 input LUTs	219	4%
Number of bonded IOBs	23	14%
Number of BUFGMUXs	6	25%
Average Fan out of Non-Clock Nets	2.48	

Table.2. Booth's Multiplier Implemented Using Behavioral Modeling Style

Logic Utilization	Used	Utilization
Number of Slices	3	0%
Number of Slice Flip Flops	5	0%
Number of 4 input LUTs	6	0%
Number of bonded IOBs	13	8%
Number of GCLKs	1	4%

Table.3. Synthesis Report of Booth's Multiplier Using Structural Modeling Style

Logic Utilization	Used	Utilization
Number of Slices	140	5%
Number of Slice Flip Flops	4	0%
Number of 4 input LUTs	258	5%
Number of bonded IOBs	43	27%
Number of GCLKs	1	4%

Table.4. Synthesis Report of 7 Bit Booth's Multiplier

IX. RESULT TABLE X.

Parameters for comparison	Percent Utilization	
	Behavioral	Micro architecture
Number of slices	1.4	0.1
Number of 4 input LUTs	1.3	0.1
Maximum operating Frequency (MHz)	249	339
Number of Slice Flip Flops	0.1	5
Number of Bonded IOBS	8	14
Number of GCLKs	4	25

Logic Utilization		% Utilization	
	Behavioural	Ripple carry adder	Carry Lookahead adder
Number of slices	100	0	41
Number of bounded IoBs	47	4	10
Number of GClks	15	4	2
Number of 4inputs LUTs	1	1	1
Path delay	15.538ns	4.368ns	2.949ns
Frequency	64.358MHz	228.93MHz	339.09MHz
Power Dissipation	0.052	0.081	0.052

XI. CONCLUSION

It is to be concluded that this presentation deals with the design approach of Booth's algorithm. Further, we have observed the simulation results of the booth multiplier. Booth multiplier is realized on Xilinx FPGA device using relevant synthesizer. From above result and analysis, keypad and LCD clock pulses are not get synchronize with each other, and unable to show result on LCD after 32 bit input through keypad.
So remedy to this we used switches for giving the inputs up to 15 bits and result of multiplication displayed on LED or 7 Segment Display. Also conclude that Structural modeling is more better than Behavioral modeling.

REFERENCES

1) V.R.Raut, P. R. Loya," FPGA Implementation of Low Power Booth Multiplier Using Radix-4 Algorithm" International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 3, Issue 8, August2014
2) Snehal R Deshmukh, Dinkar L Bhombe" High Performance Multiplier using Booth Algorithm", International Journal of Engineering Research & Technology (IJERT), Vol 3 Issue 4, April - 2014
3) A.RamaVasantha," Design and Implementation of FPGA Radix-4 Booth Multiplication Algorithm", International Journal of Research in Computer and Communication Technology, Vol 3, Issue 9, September - 2014
4) ZAKY, Hamacher, Computer Organization.
5) Spartan 3 Generation User Guide, http://www.xilinx.com/support/documentation/userguides/ug331.pdf

