Dr. Ravi Shankar Mishra,Prof. Puran Gour,Braj Bihari Soni / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 1, Issue 3, pp.905-910

Design and Implements of Booth and Robertson's multipliers algorithm on FPGA

Dr. Ravi Shankar Mishra
Head of the Department
NRI IIST, BHOPAL

Prof. Puran Gour
Assistant professor
NRI IIST, BHOPAL

Braj Bihari Soni
M.Tech. scholar
NRI IIST, BHOPAL

ABSTRACT

The Arithmetic and logical unit play an important role in digital systems. Particular, Multiplication is especially relevant instead of other arithmetic operators, such as division or exponentiation, which one is also utilized by multiplier as building blocks. 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. For faster computation, this paper compared Robertson's and Booth's algorithm in which quick and accurate performance of multiplier operation has been done. These algorithms provides high performance than other multiplication algorithms . For achieving these task, Xilinx ISE 8.1i software has been used and implemented on FPGA xc3s400pq208-5.

***Keywords*:** Booth, FPGA, VHDL, Robertson's, Multiplication, Xilinx ISE 8.1i

I. INTRODUCTION

Digital arithmetic operations are very important in the design of digital processors and application-specific systems. Arithmetic circuits form an important class of circuits in digital systems. With the remarkable progress in the very large scale integration (VLSI) circuit technology, many complex circuits, unthinkable yesterday have become easily realizable today.Algorithms that seemed impossible to implement now, have attractive implementation possibilities for the future. This means that not only the conventional computer arithmetic methods, but also the unconventional ones are worth investigation in new designs. Multiplication is especially relevant since other arithmetic operators, such as division or exponentiation, which they usually utilize multipliers as building blocks [3]. Hardware implementation of arithmetic operations has been oriented typically to use VLSI circuits. Among the arithmetic operations, the multiplication is widely used in applications such as graphics and scientific computation. Therefore, different kind of schemes has been developed. There are several algorithms for the computation of multiplication. Conventional algorithms include:

1) Booth's algorithm. 2) Robertson's algorithm.

1). BOOTH'S ALGORITHM

Requires that we can do an addition or a subtraction each iteration (not always an addition). When doing multiplication, strings of 0s in the multiplier require only shifting (no addition). When doing multiplication, strings of 1s in the multiplier require an operation only at each end. We need to add or subtract only at positions in the multiplier where there
is a transition from a 0 to a 1, or from a 1 to a 0.[12]

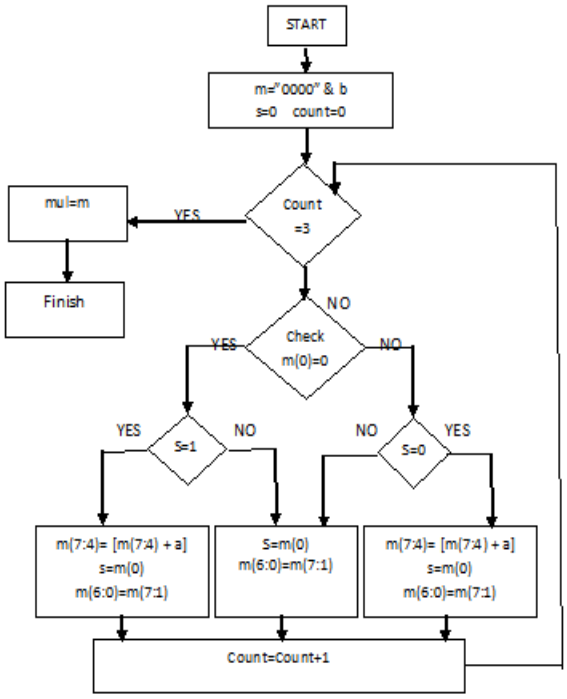

Fig1-Flowchart of booth algorithm

By above flow chart description we have, b=Multiplier, a=Multiplicand, m= Product.

First we will need twice as many bits in our product as we have in our original two operands. The leftmost bit of our operands (both multiplier and multiplicand) is a sign bit, and cannot be used as part of the value. Then Decide which operand will be the multiplier and which will be the multiplicand. If any operand and both is negative it is represent in two's complement. Begin with a product that consists of the multiplier with an additional X leading zero bits. And check the LSB and the previous LSB of product to determine the arithmetic action. If it is the FIRST pass, use 0 as the previous LSB.

Possible arithmetic actions:

00 → no arithmetic operation only shifting

01 → add multiplicand to left half of product and shifting

10 → subtract multiplicand from left half of product and shifting

11 → no arithmetic operation only shifting

So, here we are using arithmetic right shift (ASR).[14]

2).ROBERTSON'S ALGORITHM

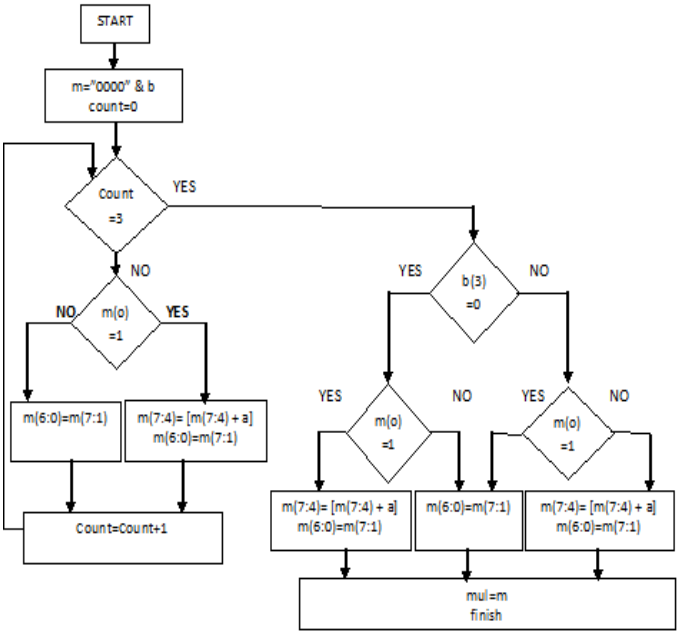

Fig2-Flowchart of Robertson's algorithm

Consider the case that we want to multiply two 8 bit numbers X = x0x1:::x3 and Y = y0y1:::y3. Depending on the sign of the two operands X and Y, there are 4 cases to be considered:

X0 = y0 = 0, that is, both X and Y are positive. Hence, multiplication of these numbers is similar to the multiplication of unsigned numbers. In other words, the product P is computed in a series of add-and-shift steps of the form

$P_i \longleftarrow P_i + X_i * Y$

$P_{i+1} \longleftarrow P_i * 2^{-1}$

Note that all partial products are non-negative. Hence, leading 0s are introduced during

right shift of the partial product.

x0 = 1; y0 = 0, that is, X is negative and Y is positive. In this case, the partial product is always positive (till the sign bit x0 is used). In the final step, a subtraction is performed. That is,

x0 = 0; y0 = 1, that is, X is positive and Y is negative. In this case, the partial product is positive and hence leading 0s are shifted into the partial product until the rest 1 in X is encountered. Multiplication of Y by this 1, and addition to the result causes the partial product to be negative, from which point on leading 1s are shifted in (rather than 0s).

x0 = 1; y0 = 1, that is, both X and Y are negative. Once again, leading 1s are shifted into the partial product whenever the partial product is negative. Also, since X is negative, the correction step (subtraction as the last step) is also performed.

Recall the difference in the correction steps between multiplication of two integers and two fractions. In the case of two integers, the correction step involves subtraction and shift right. Whereas, in the case of fractions, the correction step involves subtraction and setting.[5]

III.RESULT AND DISCUSSION

The design of fixed point 4×4 -bit booth multipliers, 4×4 -bit Robertson's multipliers has been done using VHDL and implemented in a Xilinx Spartan-3AN (Selected Device : 3s400pq208-5) FPGA using the Xilinx ISE 8.1i design tool.

Figs. 3,4,5,6 illustrate the synthesis report and RTL View (Top Module) and Internal RTL View and Simulation result of 4×4 -bit Booth multipliers .Figs. 7,8,9,10 illustrate the synthesis report and RTL View (Top Module) and Internal RTL View and Simulation result of 4×4 -bit Robertson's multipliers.

Table 1 summarizes the FPGA device resources utilization for 4×4 Booth and 4×4Robertson's multiplier for Spartan-3AN (Selected Device : 3s400pq208-5).

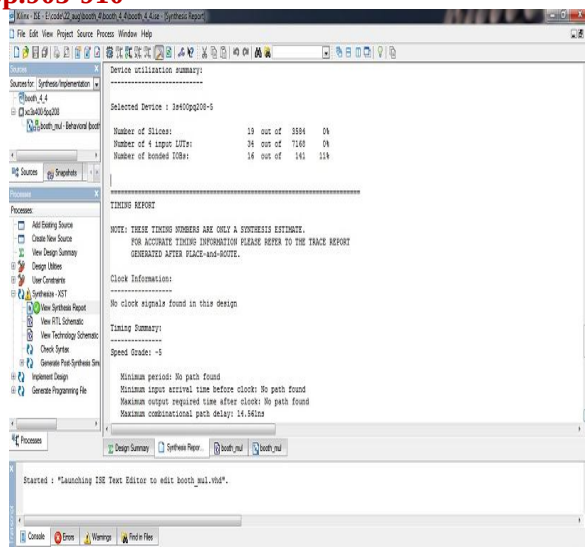

Fig.3: synthesis report of 4×4 -bit Booth multiplier

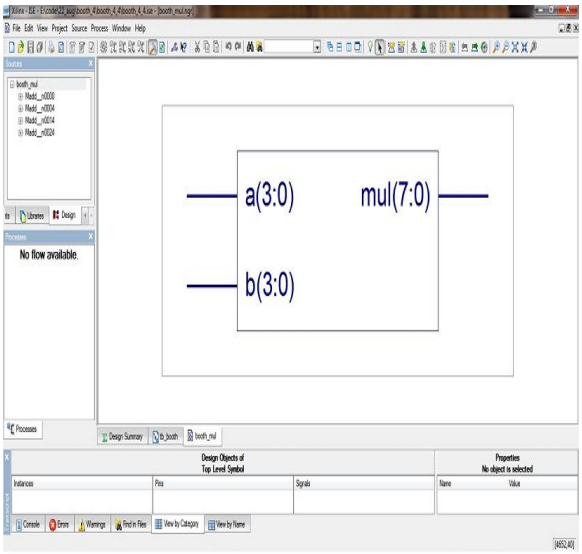

Fig.4: RTL View of 4×4 -bit Booth multiplier (Top Module)

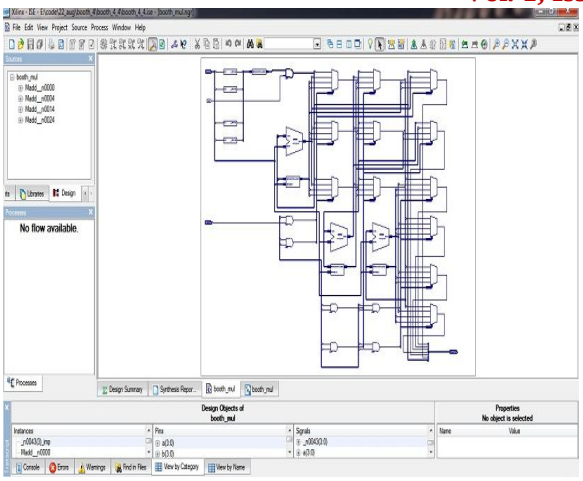

Fig 5: Internal RTL View of 4×4 -bit Booth multiplier

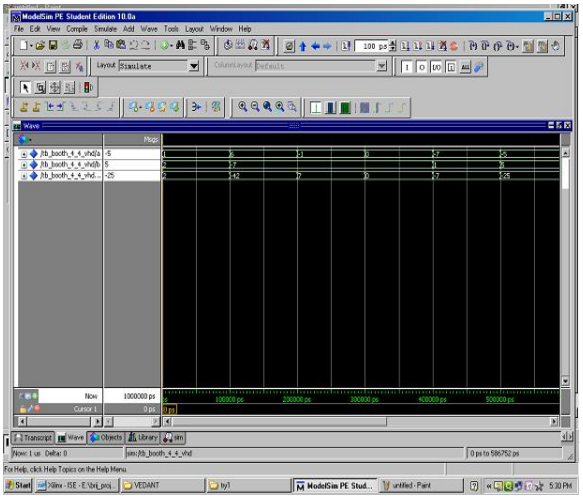

Fig 6: Simulation of 4×4 -bit Booth multiplier

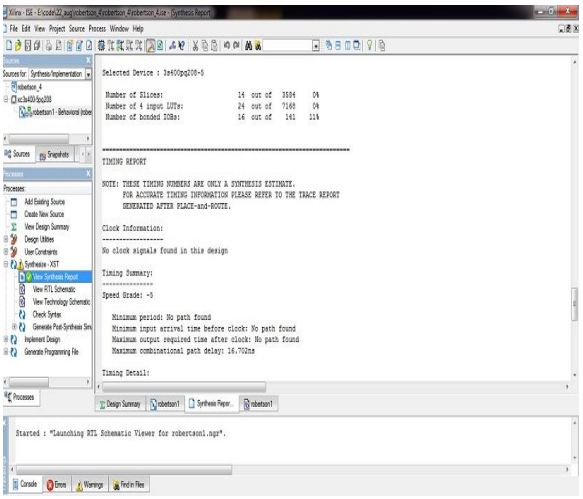

Fig.7: synthesis report of 4×4 -bit Robertson's multiplier

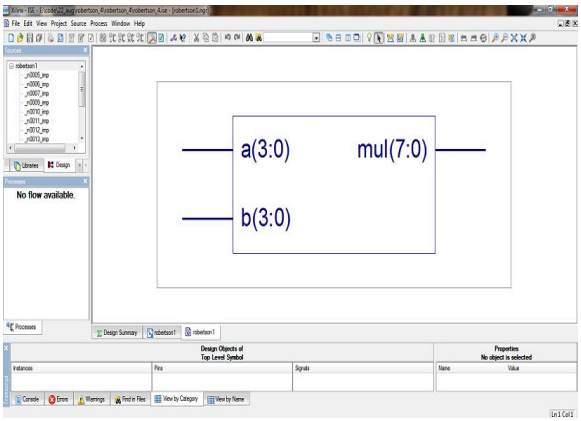

Fig.8: RTL View of 4×4 -bit Robertson's multiplier (Top Module)

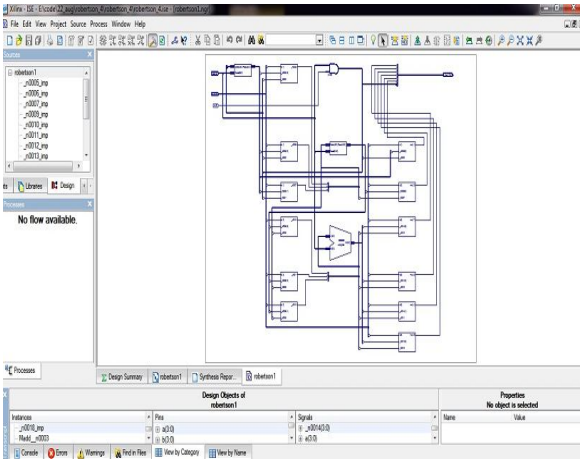

Fig.9: Internal RTL View of 4×4 -bit Robertson's multiplier

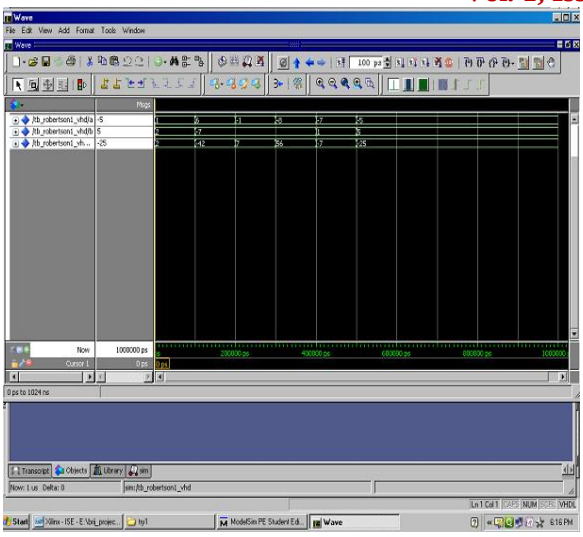

Fig.10: Simulation of 4×4 -bit Robertson's multiplier

IV. DEVICE UTILIZATION

Table 1:FPGA resource utilization for Booth and Robertson's multiplier for Spartan-3AN (Selected Device : 3s400pq208-5)

Bit Width	Multipliers algorithm	Number of Slices	Number of 4 input LUTs	Number of bonded IOBs	Dely
4×4	Booth's	19	34	16	14.56 ns
4×4	Robertson's	14	24	16	16.70 ns

V. CONCLUSION

We have presented the FPGA implementation for fixed point 4×4 Booth multiplier algorithm , 4×4 Robertson's multiplier algorithm. So, after comparison of booth algorithm and Robertson's algorithm results concluded that, Robertson's algorithm used less hardware but booth algorithm take less delay time.

We have used a FPGA Spartan XC3S400pq208-5 for physical implementation and for synthesis and simulation process we have used the computation packet ISE 8.1i provided by Xilinx.

References

[1] Muhammad H. Rais and Mohamed H. Al Mijalli, "FPGA Based Fixed Width 4×4, 6×6, 8×8 and 12×12-Bit Multipliers using Spartan-3AN", *IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.2, February 2011*

[2] T.J. Todman, G.A. Constantinides, S.J.E. Wilton, O. Mencer, W. Luk and P.Y.K. Cheung, "Reconfigurable computing: architectures and design methods", *in IEE Proc. of the Computer and Digital Techniques, 2005, Vol. 152, No. 2, pp. 193-207.*

[3] M.A.G. Martinez, R.P. Gomez, G.M. Luna and F.R. Henrique, "FPGA Implementation of an Efficient Multiplier over Finite Fields GF(2m)" *Proceedings of International Conf. On Reconfigurable Computing and FPGAs, 2005.*

[4] C. Maxfield, The Design Warrior's Guide to FPGAs: Devices, Tools and flows. Newnes Publishers, MA, 2004.

[5] John Wiley & Sons - 2004 – "Arithmetic and Logic in Computer Systems" *A JOHN WILEY & SONS, INC., PUBLICATION.*

[6] E.III. Walters, M.G. Arnold, and M.J. Schulte, "Using truncated multipliers in DCT and IDCT hardware accelerators", *in Proc. of the XIII SPIE Advanced Signal Processing Algorithms, Architectures, and Implementations, 2003, pp. 573-584.*

[7] J.H.Kim, J.S.Lee and J.D.Cho "A Low Power Booth Multiplier Based on Operand Swapping in Instruction Level" *Journal of the Korean Physical Society, Vol. 33, No. , January 1998, pp. S258_S261.*

[8] Stuart F. Oberman, and Michael J. Flynn, "Division Algorithms and Implementations" *IEEE Transactions on Computers, vol. 46, no. 8, August 1997*

[9] L. Song, K.K. Parhi, "Efficient Finite Field Serial/Parallel Multiplication", *Proc. of International Conf. On Application Specific Systems, Architectures and Processors, pp. 72-82, Chicago,USA, 1996.*

[10] Rajendra Katti "A Modified Booth Algorithm for High Radix Fixed-point Multiplication" *IEEE*

Trans. VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL 2. NO 1. Dec. l994.

[11] P. E. Madrid, B. Millar, and E. E. Swartzlander, "Modified Booth algorithm for high radix fixed-point multiplication," *IEEE Trans. VLSI Syst., vol. 1 , no. 2, pp. 164-167, June 1993*

[12] J. J. F. Cavanagh, Digital Computer Arithmetic, *New York: McGraw Hill, 1984.*

[13] Douglas L. Perry, VHDL programming by Example. [online]
16.2.2009 [web*]*
http://books.google.cz/books?id=aWWpc50wChIC&printsec=frontcover&dq=VHDL