International Journal of Engineering & Technology, 7 (2.7) (2018) 409-413

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Implementation of time efficient hybrid multiplier for FFT computation

R. Nikhil [1] *, G. V. S. Veerendra [1], J. Rahul M. S. Sri Harsha [1], Dr. V. S. V. Prabhakar [1]

[1] Koneru Lakshmaiah Educational Foundation Vaddeswaram, Guntur, Andhra Pradesh, India-522502
**Corresponding author E-mail: ebbanikhil@gmail.com*

Abstract

Now a days in designing a VLSI circuits we are coming across many problems such as high power intake, delay and large utilization of chip area in order to overcome these problems a new architectures are developed. In our project we deals with FFT computation which internally involves series of multiplication and addition therefore requirement of efficient multipliers is needed and therefore we come across two high speed improved multipliers Booth multiplier and Wallace tree multiplier which are good in terms of power efficiency and low output delay. The main aim of our project involves hybridizing the both Wallace multiplier and Booth multiplier which yields low delay and low power consumption than compared to individual multipliers. The Booth multiplier is used for reduction of partial products and for addition operations carry save adders is used in Wallace tree multipliers and thus hybrid is designed by combining both the algorithms which in turn produces better results and they can be observed in comparisons tabular column in our documentation. These multipliers can be designed in many ways such using cmos layout techniques and also using Verilog programming and we have chosen Verilog programming which requires Xilinx software and codes are developed in gate level design model for the respective multiplier models and the results will be tabulated.

Keywords: *Multiplier; Xilinx; Verilog Programming; High Speed.*

1. Introduction

In present day scenario FFT became more significant for signal processing applications. FFT involves bit reversed input and natural order output and vice-versa of these can be taken. Series of multiplication and addition operations are performed in FFT to compute the result. Here frequency coefficient can be taken into considerations [1]. DFT is an algorithm to perform convolutions or multiplication operations and is used to perform FFT in many practical applications [2].

As referred in paper [1], [2] An FFT is a successful algorithm in order to compute the DFT and IDFT. There are numerous unmistakable FFT algorithms involving an extensive variety of science, since basic complex-number math to group theory and number hypothesis. The quick Fourier Transform is a profoundly productive strategy for processing the DFT of a limited arrangement and requires less number of calculations than that of coordinate assessment of DFT. It lessens the calculations by exploiting the fact that the computation of the coefficients of the DFT can be done iteratively. Due to this, FFT calculation strategy remains utilized as a part of advanced otherworldly investigation, channel simulation, autocorrelation and example recognition. The FFT depends on disintegration and breaking the change into littler changes and consolidating them to get the aggregate change. FFT diminishes the computation time required to process a discrete Fourier change and enhances the execution by a factor of at least 100 over direct assessment of the DFT.

As referred in paper [3], [5] In this paper the main theme is that we considered the booth multiplier which uses booth encoding algorithm in order to reduce the number of incomplete items by thinking about two bits of a multiplier. The fundamental operation of this strategy is increasing both marked and unsigned numbers.

As referred in paper [4] the main theme of this paper is the fast multiplication of two numbers. A Wallace tree is an effective equipment usage of a computerized circuit that increases two numbers keeping in mind the end goal to play out the augmentation of two numbers with the Wallace strategy.

As referred in Paper [3], [4], [5] the main aim in this paper is to get information about the multipliers. The multipliers are composed into three major blocks the modified booth encoder and multiplicand sector block for formation of the partial products, the second is the Wallace tree area which includes all the halfway item numbers and deliver two numbers and last pieces comprises of the convey select viper segment which includes the two numbers obtained from the Wallace tree output As referred in paper: The main theme of this paper is to know the advantages and disadvantages of Wallace tree multiplier. The advantage of the Wallace tree is that there is an exclusive decrease layer and each layer has a spread postponement. As making the halfway items is and the last expansion is, the increase is just, very little slower than expansion.

2. Introduction to multiplier

Increase is a critical basic capacity in number-crunching rationale operation. Computational execution of a DSP framework is restricted by its duplication execution and since, increase commands the execution time of most DSP calculations; subsequently rapid multiplier is abundantly wanted. As of now, augmentation time is as yet the predominant factor in deciding the guideline process duration of a DSP chip. With a consistently expanding mission for more noteworthy processing power on battery-worked cell phones, plan accentuation has moved from improving traditional defer time region size to limiting force scattering while as yet keeping up the elite. Generally move and add calculation has been executed to outline however this isn't reasonable for VLSI usage and furthermore from postpone perspective. A portion of the critical calculation proposed in writing for VLSI implementable quick increase is Booth multiplier, cluster multiplier and Wallace tree multiplier. This paper exhibits the crucial specialized angles behind these methodologies. The low power and rapid VLSI can be actualized with various rationale styles. The three critical contemplations for VLSI configuration are power, zone and postponement. There are many proposed rationales (or) low power scattering and fast and every rationale style has its own favorable circumstances as far as speed and power.

The goal of good multiplier to give a physically smaller rapid and low power utilization unit. Being a center piece of math handling unit multipliers are in to a great degree appeal on its speed and low power utilization. To decrease noteworthy power utilization of multiplier outlines it is a decent bearing to diminish number of operations along these lines lessening a dynamic power which is a noteworthy piece of aggregate power dispersal. In the past extensive exertion were put into planning multiplier in VLSI toward this path.

There are number of procedures that to perform double augmentation. When all is said in done, the decision depends on elements, for example, dormancy, throughput, zone, and outline multifaceted nature. More effective parallel approach utilizes a type of exhibit or tree of full adders to total incomplete items. Exhibit multiplier, Booth Multiplier and Wallace Tree multipliers are a portion of the standard approaches to have hardware implementation of binary multiplier which are reasonable for VLSI execution at CMOS level.

2.1. Wallace tree multiplier

A quick procedure for duplication of two numbers was produced by Wallace. Utilizing this technique, a three stage process is utilized to increase two numbers; the bit items are shaped, the bit item framework is diminished to a two column network where whole of the line approaches the total of bit items, and the two coming about lines are summed with a quick snake to create a last item. Three piece signals are passed to a one piece full viper ("3W") which is known as a three info Wallace tree circuit, and the yield flag (aggregate flag) is provided to the following stage full snake of a similar piece, and the convey yield flag thereof is passed to the following stage full viper of the same no of bit, and the convey yield flag thereof is provided to the following phase of the full viper situated at a one piece higher position. Wallace tree is a tree of pass on save adders planned as showed up in figure. A convey spare snake comprises of full adders like the more commonplace swell adders, yet the convey yield from each piece is conveyed out to frame second outcome vector preferably being than wired to the following most noteworthy piece. The convey vector is 'spared' to be joined with the entirety later. In the Wallace tree strategy, the circuit format isn't simple despite the fact that the speed of the operation is high since the circuit is very sporadic.

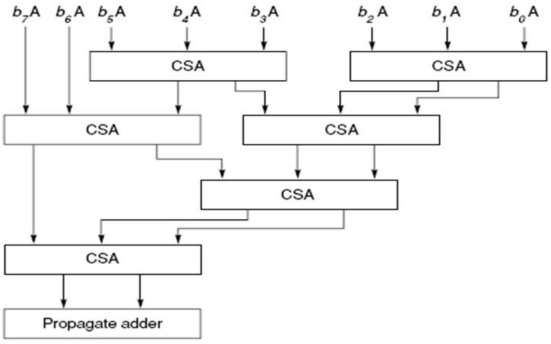


Fig. 1: Wallace Tree Multiplier.

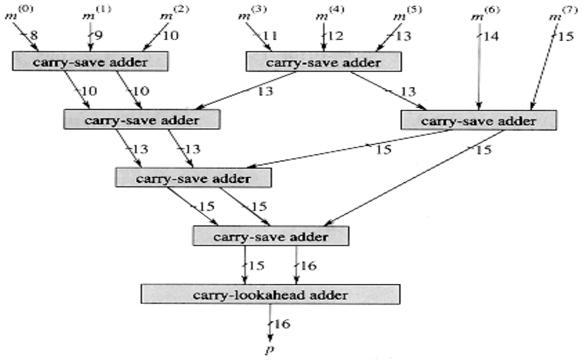


Fig. 2: An Example of Wallace Tree Multiplier.

2.2. Booth multiplier

Andrew D. Booth in 1951 in order to overcome the limitations in normal conventional array multipliers in terms of chip area and delay a new architecture is implemented and named as Booth multiplier. Booth algorithm deals with minimizing the partial products by considering at time two bits of multiplier. Therefore, achieving less delay compared to Braun multiplier and Baugh Wooley multiplier. Using this algorithm, we can compute multiplication for both signed and unsigned integers and based on Radix-2 computation it accepts the numbers in two's complement form

To quicken the expansion Booth multiplier encode plays out a couple of phases of growth immediately. Corner's computation abuses the way that a snake subtractor is about as speedy and little as a clear snake.

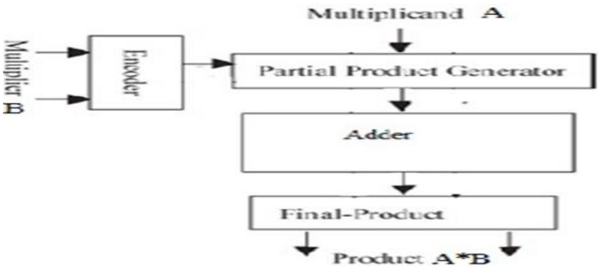
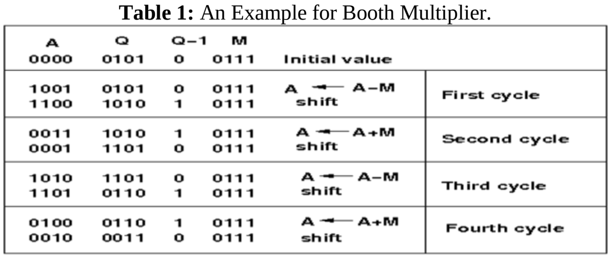


Fig. 3: Booth Multiplier.

Table 1: An Example for Booth Multiplier.



A	Q	Q−1	M		
0000	0101	0	0111	Initial value	
1001	0101	0	0111	A ← A−M	First cycle
1100	1010	1	0111	shift	
0011	1010	1	0111	A ← A+M	Second cycle
0001	1101	0	0111	shift	
1010	1101	0	0111	A ← A−M	Third cycle
1101	0110	1	0111	shift	
0100	0110	1	0111	A ← A+M	Fourth cycle
0010	0011	0	0111	shift	

2. 3. Booth-encoded Wallace tree multiplier

On studying the various types of multipliers, we have come to know that Booth multiplier and Wallace multiplier produces best results when compared to other multipliers. This multiplier is designed by hybridizing the both Wallace and Booth multiplier hence achieving the benefits of both the multipliers. Booth multiplier reduces the number of fragmented things. Carry save adder in Wallace tree multiplier performs addition of m numbers less than normal adder in less duration and it contains full adders like ripple carry adders and by use of carry save adder carry propagation is avoided in the adder.

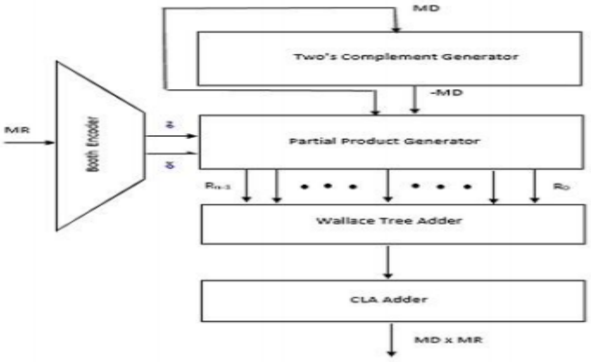


Fig. 4: Booth Encoded Wallace Tree Multiplier.

3. Experimental investigation

We have implemented the Verilog programming for the Wallace tree multiplier; booth multiplier and hybrid multiplier using Xilinx Software. Comparison between these multipliers are done and was results are tabulated.

3.1. Xilinx software

3.1.1. Verilog programming

Verilog, institutionalized as IEEE 1364, is an equipment portrayal dialect (HDL) used to show electronic frameworks. It is most normally used as a part of the summary and check of advanced circuits during the enroll exchange level of deliberation. It is similarly used as a part of the check of simple circuits and blended flag circuits, and in addition Xilinx ISE is an item contraption conveyed through Xilinx for mix and examination of HDLdesigns, engaging the creator to organize ("compile") their plans, perform timing examination, investigate RTL traces, imitate a blueprint's reaction to different shocks, and orchestrate the target device with the product build.
Xilinx ISE is a diagram circumstance for FPGA things from Xilinx, and is immovably coupled to the building of such chips, and can't be used with FPGA things from different merchants. The Xilinx ISE is on a very basic level used for circuit union and plan, while ISIM or the ModelSim method of reasoning test framework is used for system level testing. Distinctive fragments transported with the Xilinx ISE join the Embedded Development Kit (EDK), a Software Development Kit (SDK) and ChipScope Pro
Both VHDL and Verilog Programming ought to be conceivable in Xilinx Software in the outline of hereditary circuits. Verilog is a portmanteau of the words "confirmation" and "rationale".
Types of levels in Verilog programming

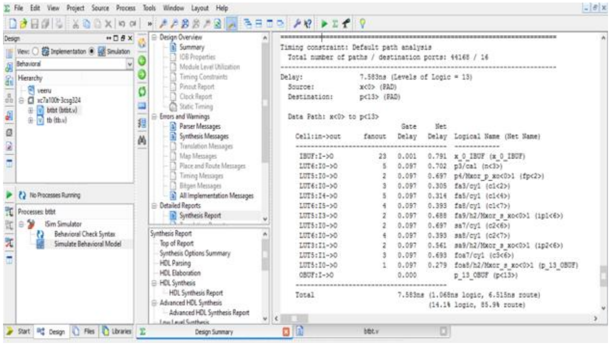
1) Behavioral level programming
2) Data level modeling
3) Gate level programming

3.1.2. Benefits

- Compact language-less code.
- Reduction operators- operate on a single operator.
- Low-level descriptions closer to actual hardware-automatically declares wire and registers.
- Compiler directives.

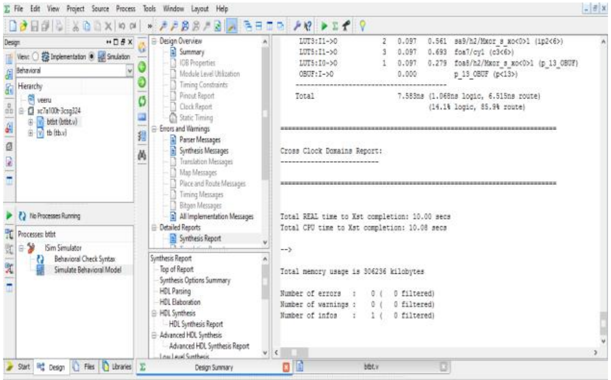
4. Experimental results

4.1. Booth encoded multiplier



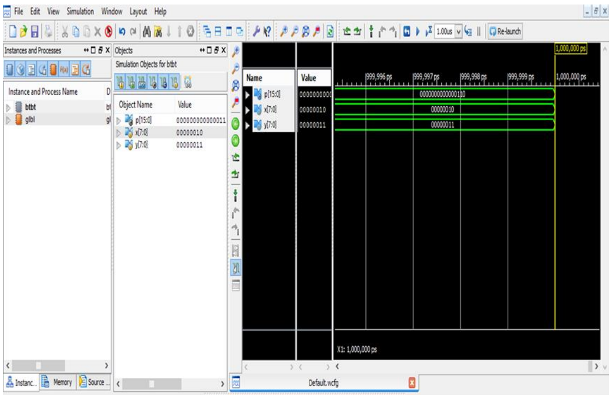
```
================================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 44168 / 16
--------------------------------------------------------------------------------
Delay:               7.583ns (Levels of Logic = 13)
  Source:            x<0> (PAD)
  Destination:       p<13> (PAD)

  Data Path: x<0> to p<13>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            23   0.001   0.791  x_0_IBUF (x_0_IBUF)
     LUT6:I0->O            5   0.097   0.702  p3/ca1 (n<3>)
     LUT5:I0->O            2   0.097   0.697  p4/Mxor_p_xo<0>1 (fp<2>)
     LUT6:I0->O            3   0.097   0.305  fa3/cy1 (c1<2>)
     LUT5:I4->O            5   0.097   0.314  fa5/cy1 (c1<4>)
     LUT6:I5->O            4   0.097   0.393  fa8/cy1 (c1<7>)
     LUT5:I3->O            2   0.097   0.688  fa9/h2/Mxor_s_xo<0>1 (ip1<6>)
     LUT5:I0->O            2   0.097   0.697  sa7/cy1 (c2<6>)
     LUT6:I0->O            4   0.097   0.393  sa8/cy1 (c2<7>)
     LUT3:I1->O            2   0.097   0.561  sa9/h2/Mxor_s_xo<0>1 (ip2<6>)
     LUT5:I1->O            3   0.097   0.693  foa7/cy1 (c3<6>)
     LUT5:I0->O            1   0.097   0.279  foa8/h2/Mxor_s_xo<0>1 (p_13_OBUF)
     OBUF:I->O                 0.000          p_13_OBUF (p<13>)
    ----------------------------------------
    Total                      7.583ns (1.068ns logic, 6.515ns route)
                                       (14.1% logic, 85.9% route)
```

Fig. 5: Delay Calculation of Booth Multiplier.



```
     LUT3:I1->O            2   0.097   0.561  sa9/h2/Mxor_s_xo<0>1 (ip2<6>)
     LUT5:I1->O            3   0.097   0.693  foa7/cy1 (c3<6>)
     LUT5:I0->O            1   0.097   0.279  foa8/h2/Mxor_s_xo<0>1 (p_13_OBUF)
     OBUF:I->O                 0.000          p_13_OBUF (p<13>)
    ----------------------------------------
    Total                      7.583ns (1.068ns logic, 6.515ns route)
                                       (14.1% logic, 85.9% route)

================================================================================

Cross Clock Domains Report:
--------------------------

================================================================================


Total REAL time to Xst completion: 10.00 secs
Total CPU time to Xst completion: 10.08 secs

-->

Total memory usage is 306236 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    0 (   0 filtered)
Number of infos    :    1 (   0 filtered)
```

Fig. 6: Memory Usage of Booth Multiplier.

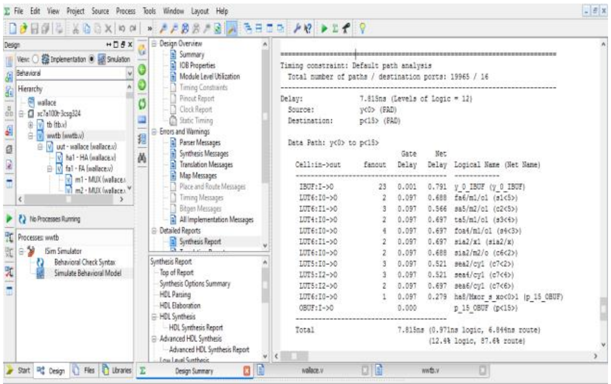


Object Name	Value
p[15:0]	00000000000011
x[7:0]	00000010
y[7:0]	00000011

Name	Value
p[15:0]	000000000
x[7:0]	00000010
y[7:0]	00000011

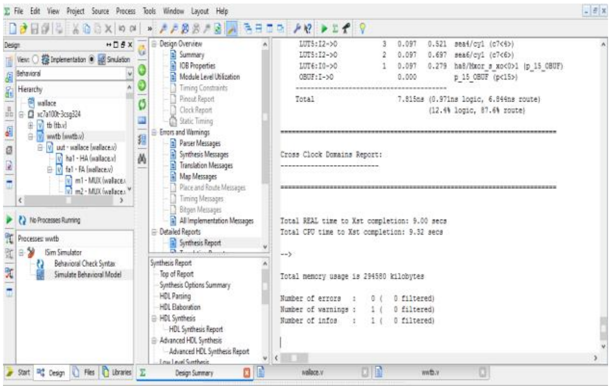
Fig. 7: Simulation Results of Booth Multiplier.

4.2. Wallace tree multiplier



```
================================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 19965 / 16
--------------------------------------------------------------------------------
Delay:               7.815ns (Levels of Logic = 12)
  Source:            y<0> (PAD)
  Destination:       p<15> (PAD)

  Data Path: y<0> to p<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            23   0.001   0.791  y_0_IBUF (y_0_IBUF)
     LUT6:I0->O            2   0.097   0.688  fa6/m1/o1 (s1<5>)
     LUT6:I1->O            3   0.097   0.566  sa5/m2/o1 (c2<5>)
     LUT4:I0->O            2   0.097   0.697  ta5/m1/o1 (s3<4>)
     LUT6:I0->O            4   0.097   0.697  foa4/m1/o1 (s4<3>)
     LUT6:I1->O            2   0.097   0.697  sia2/x1 (sia2/x)
     LUT6:I0->O            2   0.097   0.688  sia2/m2/o (c6<2>)
     LUT5:I0->O            3   0.097   0.521  sea2/cy1 (c7<2>)
     LUT5:I2->O            3   0.097   0.521  sea4/cy1 (c7<4>)
     LUT5:I2->O            2   0.097   0.697  sea6/cy1 (c7<6>)
     LUT6:I0->O            1   0.097   0.279  ha8/Mxor_s_xo<0>1 (p_15_OBUF)
     OBUF:I->O                 0.000          p_15_OBUF (p<15>)
    ----------------------------------------
    Total                      7.815ns (0.971ns logic, 6.844ns route)
                                       (12.4% logic, 87.6% route)
```

Fig. 8: Delay Calculations of Wallace Tree Multiplier.



```
 LUT5:I2->O            3   0.097   0.521  sea4/cy1 (c7<4>)
 LUT5:I2->O            2   0.097   0.697  sea6/cy1 (c7<6>)
 LUT6:I0->O            1   0.097   0.279  ha8/Mxor_s_xo<0>1 (p_15_OBUF)
 OBUF:I->O                 0.000          p_15_OBUF (p<15>)
 ----------------------------------------
Total                      7.815ns (0.971ns logic, 6.844ns route)
                                   (12.4% logic, 87.6% route)

=========================================================================

Cross Clock Domains Report:
--------------------------

=========================================================================


Total REAL time to Xst completion: 9.00 secs
Total CPU time to Xst completion: 9.32 secs

-->

Total memory usage is 294580 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    1 (   0 filtered)
Number of infos    :    1 (   0 filtered)
```

Fig. 9: Memory Usage of Wallace Tree Multiplier.

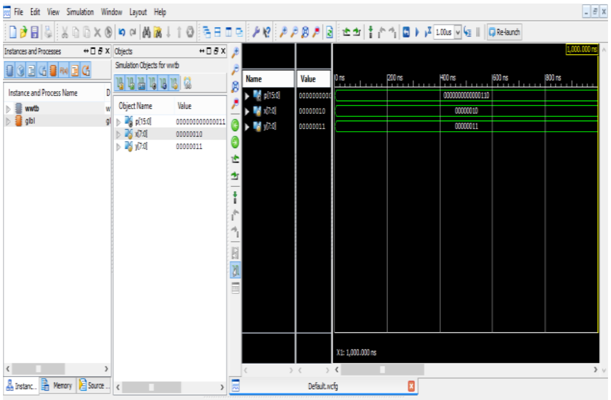
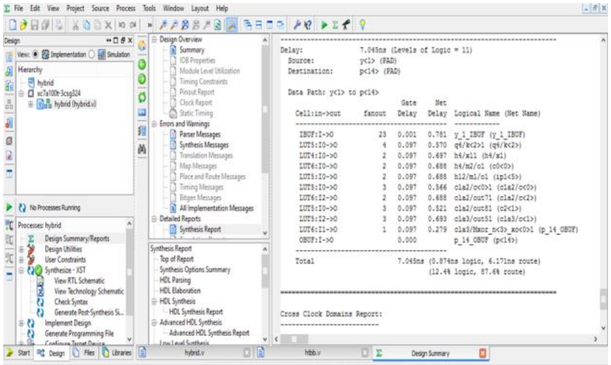


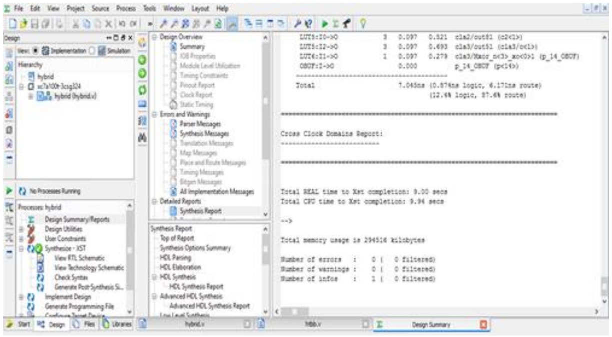
Fig. 10: Simulation Results of Wallace Tree Multipliers.

4.3. Booth encoded Wallace tree (hybrid) multiplier



```
-------------------------------------------------------------------------
Delay:               7.045ns (Levels of Logic = 11)
  Source:            y<1> (PAD)
  Destination:       p<14> (PAD)

  Data Path: y<1> to p<14>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            23   0.001   0.781  y_1_IBUF (y_1_IBUF)
     LUT5:I0->O            4   0.097   0.570  q4/k<2>1 (q4/k<2>)
     LUT4:I0->O            2   0.097   0.697  h4/x11 (h4/x1)
     LUT6:I0->O            2   0.097   0.688  h4/m2/o1 (c0<0>)
     LUT5:I0->O            2   0.097   0.688  h12/m1/o1 (ip1<5>)
     LUT5:I0->O            3   0.097   0.566  cla2/c<0>1 (cla2/c<0>)
     LUT6:I2->O            2   0.097   0.688  cla2/out71 (cla2/c<2>)
     LUT5:I0->O            3   0.097   0.521  cla2/out81 (c2<1>)
     LUT5:I2->O            3   0.097   0.693  cla3/out51 (cla3/c<1>)
     LUT6:I1->O            1   0.097   0.279  cla3/Mxor_n<3>_xo<0>1 (p_14_OBUF)
     OBUF:I->O                 0.000          p_14_OBUF (p<14>)
    ----------------------------------------
    Total                      7.045ns (0.874ns logic, 6.171ns route)
                                       (12.4% logic, 87.6% route)

=========================================================================

Cross Clock Domains Report:
--------------------------
```

Fig. 11: Delay Calculations of Booth Encoded Wallace Tree Multiplier.



```
     LUT5:I0->O            3   0.097   0.521  cla2/out81 (c2<1>)
     LUT5:I2->O            3   0.097   0.693  cla3/out51 (cla3/c<1>)
     LUT6:I1->O            1   0.097   0.279  cla3/Mxor_n<3>_xo<0>1 (p_14_OBUF)
     OBUF:I->O                 0.000          p_14_OBUF (p<14>)
    ----------------------------------------
    Total                      7.045ns (0.874ns logic, 6.171ns route)
                                       (12.4% logic, 87.6% route)

=========================================================================

Cross Clock Domains Report:
--------------------------

=========================================================================


Total REAL time to Xst completion: 9.00 secs
Total CPU time to Xst completion: 9.94 secs

-->

Total memory usage is 294516 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    0 (   0 filtered)
Number of infos    :    1 (   0 filtered)
```

Fig. 12: Memory Usage of Booth Encoded Wallace Tree Multiplier.

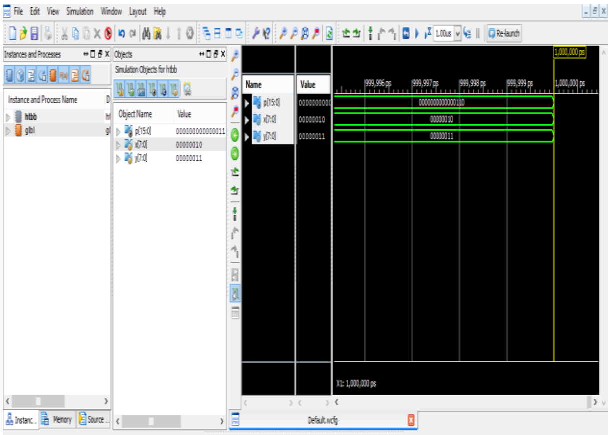


Fig. 13: Simulation Results of Booth Encoded Wallace Tree Multiplier.

5. Discussion of results

As from the above figures obtained through the simulation of Wallace tree booth multiplier and hybrid multiplier we came to know that for a taken input they all three provided an output but each multiplier with different delays and power consumption as shown in design summary according to the programs written to the particular multipliers depending and number of gates used in the program and efficiency of the code will decide the optimization for parameters like chip delay power consumption and area In Wallace tree multiplier along with partial product minimization code we have also written program for carry save adder so the weight of the code is more compared to booth multiplier hence among both the improved multipliers booth multiplier is most efficient one which produces optimized outputs and finally coming to hybrid both the multipliers booth and Wallace are combined and an efficient code is generated with much more optimized output parameters. The entire delay of the three programs for multipliers depends on number of input buffers lookup tables and output buffers usage by the respective block diagrams generated.

Table 2: Delay Comparison of Different Multipliers

S.NO	Multiplier	Delay(ns)
1	Booth Multiplier	7.58
2	Wallace Tree Multiplier	7.81
3	Booth encoded Wallace tree multiplier	7.04

6. Conclusion

This project mainly deals about performance of different types of multipliers, normally multipliers are used in many parts of digital devices such as computers, microprocessors and microcontrollers and also in signal analysis such as FFT computation which involves series of multiplications and additions. Fast Fourier transform (FFT) is used to compute signal analysis algorithms and here to enhance the speed of FFT operations multipliers with low delay are designed and many studies are undergoing to design an efficient multiplier for FFT computation. In our project we have studied and referred various papers and it has been found that Booth and Wallace tree are efficient among existing multipliers with respect to optimum delay and power consumption. Booth multiplier is one of the convictional array multiplier in which we can minimize intermediate products by considering at time two bits of multiplier there by achieving low delay over other multipliers and its algorithm is valid for both positive and negative integers on other hand Wallace multiplier which was developed by CS Wallace in 1964 and it involves parallel multiplication and reduction of partial products using carry save adders. Carry save adder perform addition of m numbers less than normal adder in less duration and it contains full adders like ripple carry adders and by use of carry save adder carry propagation is avoided in the adder and finally new architecture design is developed by hybridizing the both Wallace multiplier and Booth multiplier for better performance and to attain less delay.

References

[1] Mokhtar Aboelaze, Member, IEEE, “An FPGA Based Low Power Multiplier for FFT in OFDM Systems using Precomputations”, 2013 International Conference on ICT Convergence (s).

[2] Leif Sornmo, Pablo Laguna, “Bioelectrical Signal Processing in Cardiac and Neurological Applications, Copyright (c) 2005, Elsevier Inc.

[3] Sukhmeet Kaur1, Suman2 and Manpreet Singh Manna3, “Implementation of Modified Booth Algorithm (Radix 4) and its Comparison with Booth Algorithm (Radix-2)”, Advance in Electronic and Electric Engineering, Volume 3, Number 6 (2013), pp. 683-690.

[4] Rahul D Kshirsagar, Aishwarya.E.V., Ahire Shashank Vishwanath, P Jayakrishnan, “Implementation of Pipelined Booth Encoded Wallace Tree Multiplier Architecture, International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), 2013.

[5] Deepali Chandel, Gagan Kumawat, Pranay Lahoty, Vidhi Vart Chandrodaya, Shailendra Sharma, “Booth Multiplier: Ease of Multiplication”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 3, March 2013.