Design and Simulation of Radix-8 Booth Encoder Multiplier for Signed and Unsigned Numbers

Minu Thomas
M. Tech
Electronics & Communication Engineering (VLI&ES)

Abstract

The multiplication operation is present in many parts of a digital system or digital computer, most notably in signal processing, graphics and scientific computation. With advances in technology, various techniques have been proposed to design multipliers, which offer high speed, low power consumption and lesser area. Thus making them suitable for various high speeds, low power compact VLSI implementations. These three parameters i.e. power, area and speed are always traded off. This thesis work is devoted for the design and simulation of Radix-8 Booth Encoder multiplier for signed-unsigned numbers. The Radix-8 Booth Encoder circuit generates n/3 the partial products in parallel. By extending sign bit of the operands and generating an additional partial product the signed of unsigned Radix-8 Booth Encoder multiplier is obtained. The Carry Save Adder (CSA) tree and the final Carry Look ahead (CLA) adder used to speed up the multiplier operation. Since signed and unsigned multiplication operation is performed by the same multiplier unit the required hardware and the chip area reduces and this in turn reduces power dissipation and cost of a system. Verilog coding of multiplier for signed and unsigned numbers using Radix-4 booth encoder and Radix-8 booth encoder for 8X8 bit multiplication and their FPGA implementation by Xilinx Synthesis Tool on Spartan 3 kit have been done. The output has been displayed on LED of Spartan 3 kit.

Keywords: Radix-8 Booth Encoder, SUBE Multiplier, Partial Product Generator, Wallace Tree Adder

I. DESIGN OF SUBE MULTIPLIER

Multiplier architecture comprise of two architectures, i.e., Modified Booth and Wallace tree. Based on the study of various multiplier architectures, we find that Modified Booth increases the speed because it reduces partial products to half. Further, the delay in multiplier can be reduced by using Wallace tree. Power consumption of Wallace tree multiplier is also less as compared to booth and array. Features of both multipliers can be combined to produce high speed and low power multiplier.

Modified Booth multiplier consists of Modified Booth Recorder (MBR). MBR have two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The basic operation of BE is to decode the multiplier signal and output will be used by BS to generate the partial product. The partial products are then, added with the Wallace tree adders, similar to the carry save adder approach. The last row of carry and sum output is added together by carry look- ahead adder with the carry skewed to the left by position.

A. Multiplier architecture

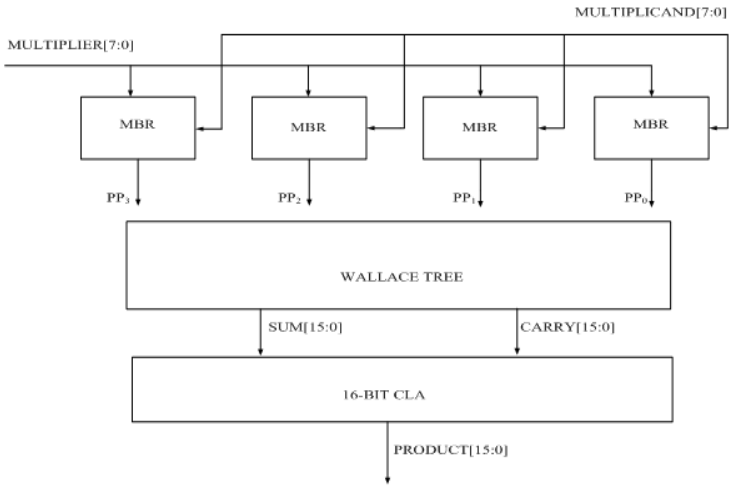

Fig. 1: Block diagram of Booth Encoded Wallace Tree multiplier

Multiplier architecture consists of MBR, Wallace tree (4:2 compressor) and a carry look- ahead adder. The first block is the new modified booth algorithm and Booth is one such algorithm which is based on the fact that fewer partial products have to be generated for groups of consecutive zeros and one's. For consecutive groups of zero's and one's, there is no need to generate any new partial product. We only need to shift the previously accumulated partial product one bit position to the right for every zero in the multiplier. In this, three consecutive bits of the multiplier are examined to find the PP's. Partial products so formed are added using 4:2 compressors. An adder that uses 4:2 compressors has a more regular structure. Wallace tree include half adders, full adders, 4:2 carry save adders in the same stage, reducing partial product at the same time. Block diagram of Booth Encoded Wallace Tree multiplier is shown in Figure 1. The outputs of Wallace tree i.e. final sum and carry are added using carry look-ahead adder and adder give the final product.

B. Booth Encoder Multiplier for Signed and Unsigned Numbers

The architecture of the proposed multiplier is shown in Fig 2. It consists of four major modules: Booth encoder, partial product generator, Wallace tree and carry look-ahead adder. The Booth encoder performs Radix-2 or Radix-4encoding of the multiplier bits. Based on the multiplicand and the encoded multiplier, partial products are generated by the generator. For large multipliers of 32 bits, the performance of the modified Booth algorithm is limited. So Booth recoding together with Wallace tree structures have been used in the proposed fast multiplier. The partial products are supplied to Wallace Tree and added appropriately. The results are finally added using a Carry Look-ahead Adder (CLA) to get the final product.

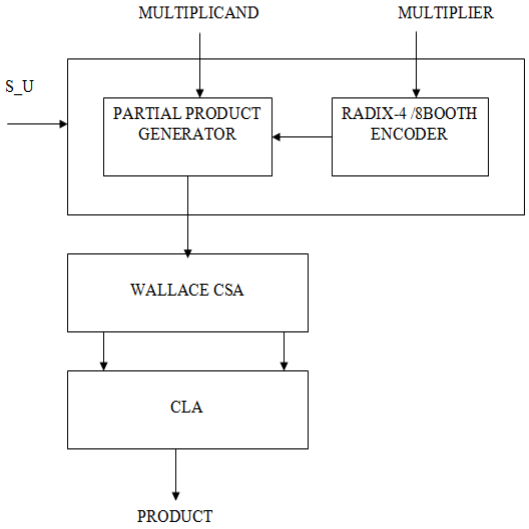

Fig. 2: Block diagram of Booth Encoder multiplier for signed unsigned numbers

1) Radix-4 Booth Encoder

Booth algorithm is a powerful algorithm for signed number multiplication, which treats both positive and negative numbers uniformly. Since a k-bit binary number can be interpreted as k/2-digit Radix-4 number, a k/3-digit Radix-8number and so on, it can deal with more than one bit of the multiplier in each cycle by using high radix multiplication. The major disadvantage of the Radix-2 algorithm was that the process required n shifts and an average of n/2 additions for an n bit multiplier. This variable number of shift and add operations is inconvenient for designing parallel multipliers .Also the Radix-2 algorithm becomes inefficient when there are isolated 1's. The Radix-4 modified Booth algorithm overcomes all these limitations of Radix-2 algorithm. For operands equal to or greater than 16 bits, the modified Radix-4Booth algorithm has been widely used. It is based on encoding the two's complement multiplier in order to reduce the number of partial products to be added to n/2.

Table I shows the encoding of the signed multiplier Y, using the Radix-4 Booth algorithm. Radix-4 Booth recoding encodes multiplier bits into [-2, 2]. Here we consider the multiplier bits in blocks of three, such that each block overlaps the previous block by one bit. It is advantageous to begin the examination of the multiplier with the least significant bit. The overlap is necessary so that we know what happened in the last block, as the most significant bit of the block acts like a sign bit.

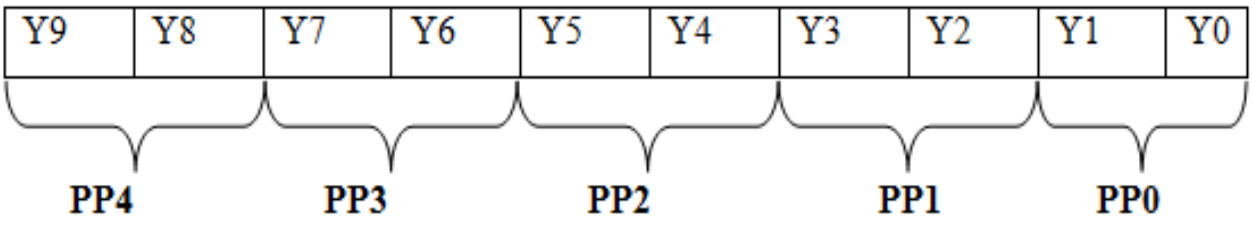

Fig.3: Grouping of bits in Radix-4 method

Table – 1
Radix - 4 Booth Recoding

Multiplier Bits Y_{2i-1} Y_{2i} Y_{2i+1}	Recoded Operation on multiplicand, X
0 0 0	0X
0 0 1	+X
0 1 0	+X
0 1 1	+2X
1 0 0	-2X
1 0 1	-X
1 1 0	-X
1 1 1	0X

2) Radix 8 Booth encoder

Radix-8 Booth recoding applies the same algorithm as that of Radix-4, but now we take quartets of bits instead of triplets. Each quartet is codified as a signed digit using Table II. Radix-8 algorithm reduces the number of partial products to n/3, where n is the number of multiplier bits. Thus it allows a time gain in the partial products summation**.**

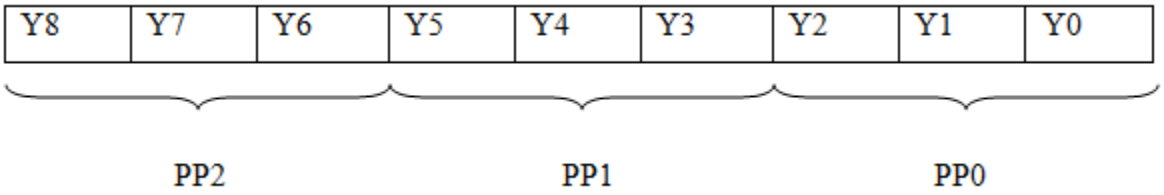

Fig. 4: Grouping of bits in Radix-8 method

Table-2
Radix -8 Booth Recoding

Multiplier Bits Y_{i+2} Y_{i+1} Y_i Y_{i-1}	Recoded Operation on multiplicand, X
0 0 0 0	0X
0 0 0 1	+X
0 0 1 0	+X
0 0 1 1	+2X
0 1 0 0	+2X
0 1 0 1	+3X
0 1 1 0	+3X
0 1 1 1	+4X
1 0 0 0	-4X
1 0 0 1	-3X
1 0 1 0	-3X
1 0 1 1	-2X
1 1 0 0	-2X
1 1 0 1	-1X
1 1 1 0	-1X
1 1 1 1	0X

3) Sign Extension Corrector

Sign Extension Corrector is designed to enhance the ability of the booth multiplier to multiply not only the unsigned number but as well as the signed number. The working principle of sign extension that converts signed multiplier signed unsigned multiplier as follows. One bit control signal called signed-unsigned(s_u) bit is used to indicate whether the multiplication operation is signed number or unsigned number .when sign-unsign s_u=0, it indicates unsigned number multiplication and when s_u=1, it indicates signed number multiplication.

Table- 3
SUBE Operation

Sign-unsign	Type of operation
0	Unsigned multiplication
1	Signed multiplication

4) Partial Product Generator

A product formed by multiplying the multiplicand by one digit of the multiplier when the multiplier has more than one digit. Partial products are used as intermediate steps in calculating larger products.

Partial product generator is designed to produce the product by multiplying the multiplicand A by 0, 1, -1, 2, -2,-3,-4, 3, 4. For product generator, multiply by zero means the multiplicand is multiplied by "0".Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place. . Multiply by "-4" is to shift left two bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by two place. Here we have an odd multiple of the multiplicand, 3Y, which is not immediately available. To generate it we need to perform this previous add: 2Y+Y=3Y. But we are designing a multiplier for specific purpose and thereby the multiplicand belongs to a previously known set of numbers which are stored in a memory chip. We have tried to take advantage of this fact, to ease the bottleneck of the radix-8 architecture, that is, the generation of 3Y. In this manner we try to attain a better overall multiplication time, or at least comparable to the time we could obtain using radix-4 architecture (with the additional advantage of using a less number of transistors). To generate 3Y with 8-bit words we only have to add 2Y+Y, that is, to add the number with the same number shifted one position to the left.

5) Wallace Tree Adder

The Wallace tree method is used in high speed designs in order to produce two rows of partial products that can be added in the last stage. Also critical path and the number of adders get reduced when compared to the conventional parallel adders. Here the Wallace tree has taken the role of accelerating the accumulation of the partial products. Its advantage becomes more pronounced for multipliers of greater than 16 bits .The speed, area and power consumption of the multipliers will be in direct proportion to the efficiency of the compressors. The Wallace tree structure with 3:2 compressors and 4:2 compressors is shown in Fig.2 and Fig.3 respectively. In this regard, we can expect a significant reduction in computing multiplications.

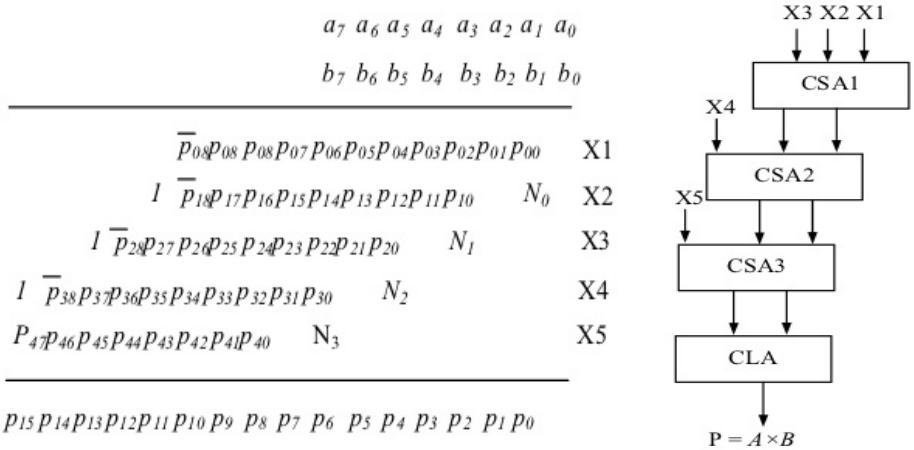

Fig. 5: Partial product adder logic ofradix-4 Booth Encoder multiplier for signed unsigned numbers

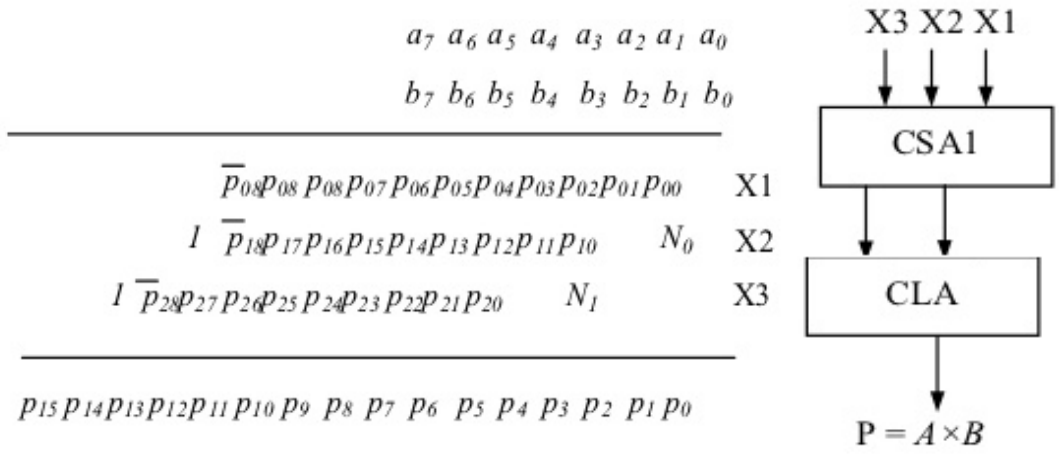

Fig. 6: Partial product adder logic ofradix-8 Booth Encoder multiplier for signed unsigned numbers

Partial products are added by the carry save adder (CSA) and the final stage is carry look ahead (CLA) adder. CSA adder takes three inputs and produce sum and carry parallel. There is one CSA. Three partial products are added by the CSA tree and finally when there are only two outputs. Left out then finally CLA adder is used to produce final result.

In all multiplication operation product is obtained by adding partial products. thus the final speed of the multiplier circuit depends on the speed of the adder circuit and the number of partial products generated .radix-8 booth encoded technique used then there are only 3 partial products and only one CSA and CLA is required to produce the final product

II. Simulation Results of 8 X8 Bit radix-4 signed unsigned Multiplier

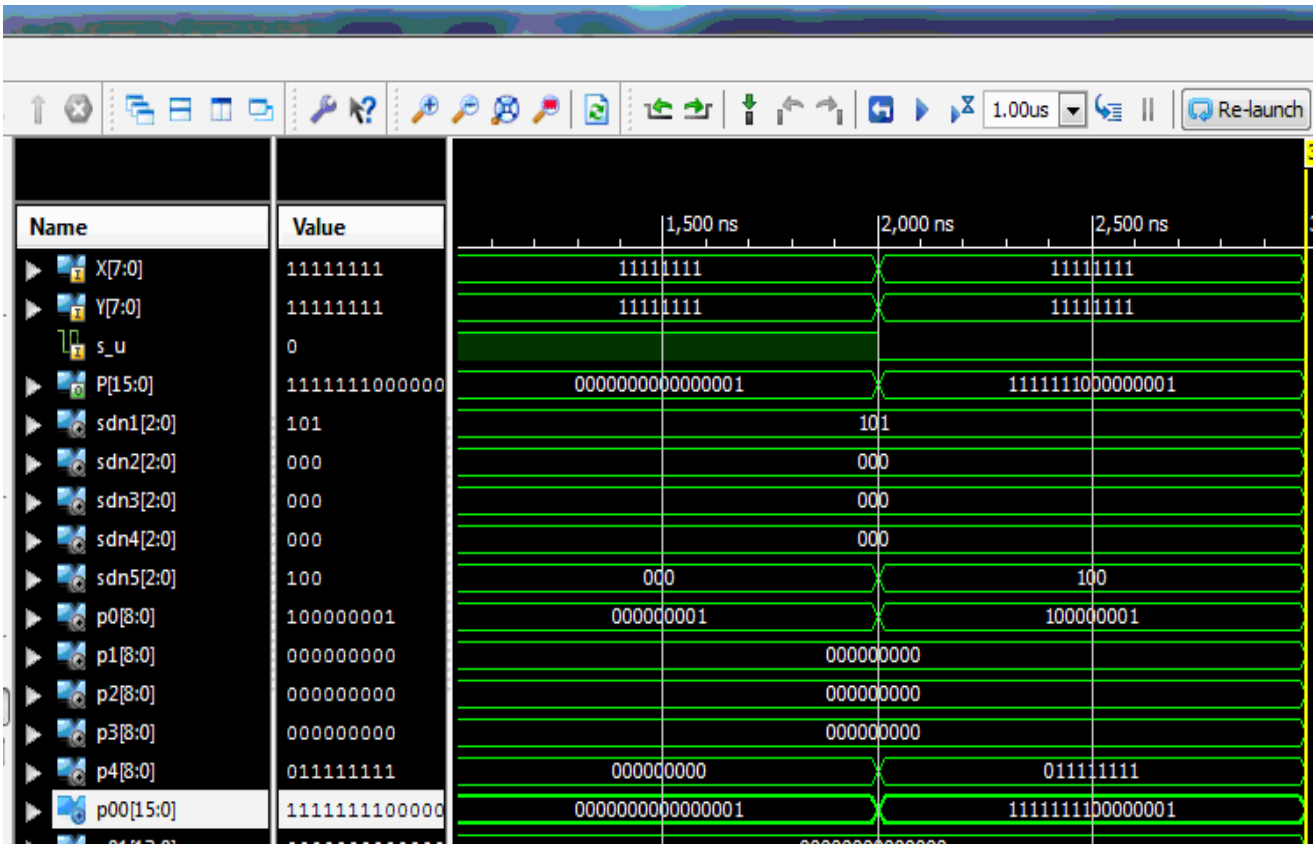

Fig. 7: Simulation Results of 8 X 8 Bit radix-4 signed unsigned Multiplier

Description: x : Input data 8 – bit, y : Input data 8– bit, p: Output data 16 – bit, x = 11111111, y = 11111111, s_u=0, p= 1111111000000001, x = 11111111, y = 11111111, s_u=1, p= 1111111111111111

Table – 4
shows Device utilization summary of 8x8 radix-4 signed unsigned booth multiplier

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	164	1,536	10%	
Number of occupied Slices	92	768	11%	
Number of Slices containing only related logic	92	92	100%	
Number of Slices containing unrelated logic	0	92	0%	
Total Number of 4 input LUTs	165	1,536	10%	
Number used as logic	164			
Number used as a route-thru	1			
Number of bonded IOBs	33	124	26%	
Average Fanout of Non-Clock Nets	3.45			

A. Timing Report of of 8 X8 signed unsigned booth multiplier

```
Source:            X<1> (PAD)
Destination:       P<15> (PAD)

Data Path: X<1> to P<15>
                              Gate     Net
  Cell:in->out      fanout    Delay    Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O            20    0.715    1.483  X_1_IBUF (X_1_IBUF)
   LUT3:I1->O            9    0.479    1.014  b2/Mrom_SDN11 (b2/Mrom_SDN)
   LUT3:I2->O            2    0.479    0.804  d2/bdo_mux0000<1>11 (d2/N6)
   LUT4:I2->O            1    0.479    0.000  d2/bdo_mux0000<2>1 (d2/bdo_mux0000<2>)
   LD:D->Q               3    0.410    1.066  d2/bdo_2 (d2/bdo_2)
   LUT3:I0->O            2    0.479    1.040  c1/f01/carry1 (c1/cc)
   LUT4:I0->O            2    0.479    1.040  c1/h5/carry1 (c1/cc1)
   LUT4:I0->O            2    0.479    0.915  c1/h8/Mxor_sum_Result1 (c1/se2)
   LUT4:I1->O            2    0.479    0.804  c1/f27/carry1 (c1/c2)
   LUT3:I2->O            2    0.479    0.804  c1/f28/carry1 (c1/c3)
   LUT3:I2->O            2    0.479    0.804  c1/f29/carry1 (c1/c4)
   LUT3:I2->O            2    0.479    0.804  c1/f30/carry1 (c1/c5)
   LUT3:I2->O            2    0.479    0.804  c1/f31/carry1 (c1/c6)
   LUT3:I2->O            2    0.479    0.804  c1/f32/carry1 (c1/c7)
   LUT3:I2->O            2    0.479    0.804  c1/f33/carry1 (c1/c8)
   LUT3:I2->O            2    0.479    0.768  c1/f34/carry1 (c1/c9)
   LUT4:I3->O            1    0.479    0.851  c1/f36/h1/Mxor_sum_Result21 (c1/N4)
   LUT3:I1->O            1    0.479    0.681  c1/f36/h2/Mxor_sum_Result1 (P_15_OBUF)
   OBUF:I->O                  4.909          P_15_OBUF (P<15>)
  ----------------------------------------
  Total                      28.994ns (13.698ns logic, 15.296ns route)
```

B. Power Report of signed unsigned booth multiplier

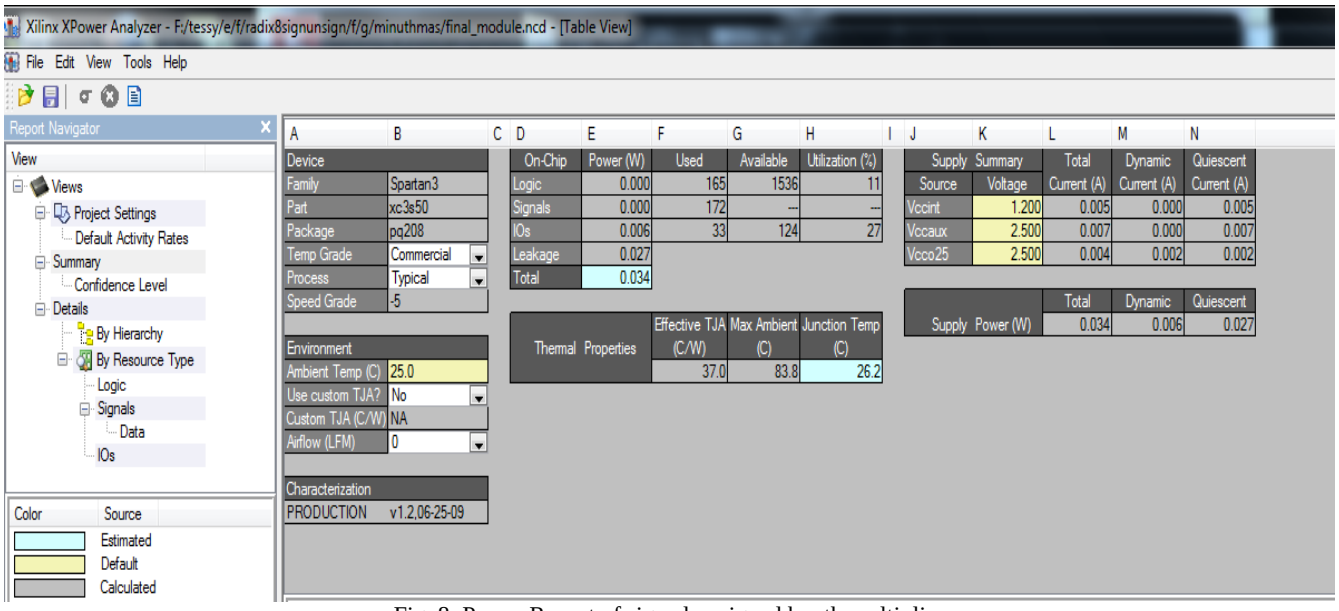

Fig. 8: Power Report of signed unsigned booth multiplier

III. SIMULATION RESULTS OF 8 X8 BIT RADIX-8 SIGNED UNSIGNED MULTIPLIER

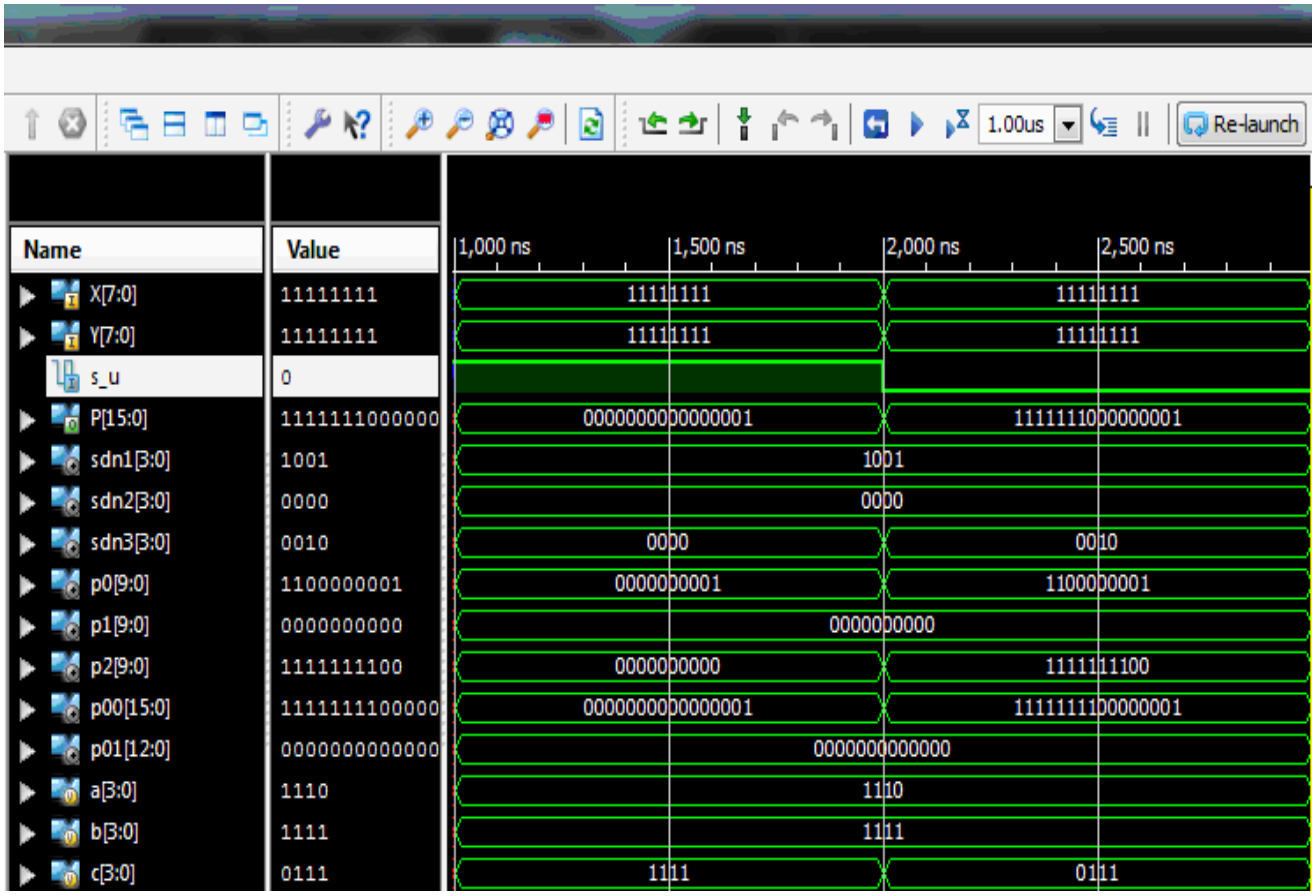

Fig. 9: Simulation Results of 8 X8 Bit radix-8 signed unsigned Multiplier

Description: x : Input data 8 – bit, y : Input data 8– bit , p: Output data 16 – bit , x = 11111111, y = 11111111, s_u=0, p= 1111111000000001, x = 11111111, y = 11111111, s_u=1, p= 1111111111111111

Table – 5
Device Utilization Summary of 8x8 Signed Booth Multiplier

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Latches	9	1,536	1%	
Number of 4 input LUTs	157	1,536	10%	
Number of occupied Slices	83	768	10%	
Number of Slices containing only related logic	83	83	100%	
Number of Slices containing unrelated logic	0	83	0%	
Total Number of 4 input LUTs	158	1,536	10%	
Number used as logic	157			
Number used as a route-thru	1			
Number of bonded IOBs	31	124	25%	
Average Fanout of Non-Clock Nets	3.68			

A. Timing Report of of 8 X8 Bit radix-8 signed unsigned Multiplier

```
Source:          Y<1> (PAD)
Destination:     P<15> (PAD)

Data Path: Y<1> to P<15>
                           Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             18   0.715   1.499  Y_1_IBUF (Y_1_IBUF)
   LUT3:I0->O             2   0.479   0.768  d3/Madd__old_A2_5_cy<2>11 (d1/Madd__old_A2_5_cy
   LUT4:I3->O             7   0.479   0.929  d3/Madd__old_A2_5_cy<4>1 (d1/Madd__old_A2_5_cy<
   LUT4:I3->O             4   0.479   1.074  d3/Madd__old_A2_5_xor<6>11 (d3/_old_A2_5<6>)
   LUT4:I0->O             3   0.479   0.830  d3/Madd__old_A3_6_cy<6>11 (d1/Madd__old_A3_6_cy
   LUT4:I2->O             3   0.479   0.941  d3/Madd__old_A3_6_xor<8>11 (d3/_old_A3_6<8>)
   LUT4:I1->O             1   0.479   0.000  d1/bdo_mux0000<8>164_SW1_G (N134)
   MUXF5:I1->O            1   0.314   0.851  d1/bdo_mux0000<8>164_SW1 (N103)
   LUT3:I1->O             1   0.479   0.000  d1/bdo_mux0000<8>164 (d1/bdo_mux0000<8>)
   LD:D->Q                3   0.410   0.941  d1/bdo_8 (d1/bdo_8)
   LUT3:I1->O             2   0.479   1.040  c1/f29/carry1 (c1/c4)
   LUT4:I0->O             5   0.479   0.842  c1/f30/carry1 (c1/c5)
   LUT4:I2->O             2   0.479   0.768  c1/f32/carry_SW1 (N13)
   LUT4:I3->O             4   0.479   1.074  c1/f36/h2/Mxor_sum_Result11 (c1/N111)
   LUT3:I0->O             1   0.479   0.000  c1/f36/h2/Mxor_sum_Result1 (c1/f36/h2/Mxor_sum_
   MUXF5:I1->O            1   0.314   0.681  c1/f36/h2/Mxor_sum_Result_f5 (P_15_OBUF)
   OBUF:I->O                  4.909          P_15_OBUF (P<15>)
  ----------------------------------------
  Total                     24.650ns (12.410ns logic, 12.240ns route)
                                     (50.3% logic, 49.7% route)
```

B. Power Report of of 8 X8 Bit radix-8 signed unsigned Multiplier

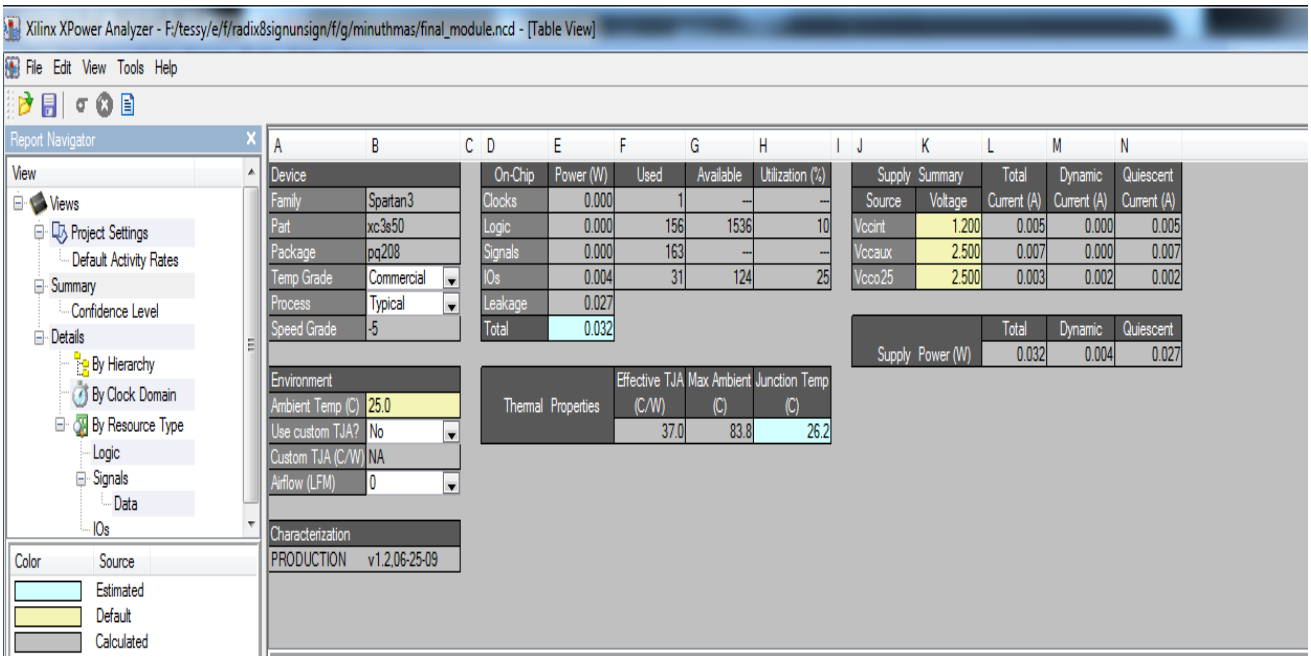

Table - 6
Comparison of 8X8 bit multipliers for Various Performance Measures

	Array multiplier	**Booth multiplier**	**Radix-4 SUBE multiplier**	**Radix-8 SUBE multiplier**
Area(LUTs)	139	140	165	158
Delay(ns)	27.952	28.171	28.994	24.650
Power(mW)	62	30	34	32

IV. CONCLUSION

The designs of 8X8 -bit, Radix-4 and Radix-8 Booth encoder multiplier for signed and unsigned numbers have been simulated using xilinx 13.3 platforms and implemented on Spartan XC3S50-5-PQ208.

The Radix-8 Booth Encoder circuit generates n/3 the partial products in parallel. By extending sign bit of the operands and generating an additional partial product the signed of unsigned Radix-8 Booth Encoder multiplier is obtained. The Carry Save Adder (CSA) tree and the final Carry Look ahead (CLA) adder used to speed up the multiplier operation. Since signed and unsigned multiplication operation is performed by the same multiplier unit the required hardware and the chip area reduces and this in turn reduces power dissipation and cost of a system.

V. FUTURE SCOPE

As an attempt to develop, arithmetic algorithm and architecture level optimization techniques for low power high-speed multiplier design, techniques presented in this thesis has achieved good results. However, there are limitations in this work and several future research directions are possible as follows:

- In order to enhance the performance, higher order compressors like7:2, 9:2 can be used to accumulate the partial products.
- Deep level pipeline architecture can be used for speed improvements.

The ability to construct very small high performance multipliers provides many other interesting possibilities. Multiplication intensive applications, such as DSP or graphics, could benefit significantly from several high performance multipliers on the same chip. A single very high throughput multiplier, or several multipliers working in parallel on the same chip, could open up new possibilities such as single chip video signal processors.

REFERENCES

[1] W. –C. Yeh and C. –W. Jen, "High Speed Booth encoded Parallel Multiplier Design," IEEE transactions on computers, vol. 49, no. 7, pp. 692-701, July 2000.

[2] Shiann-Rong Kuang, Jiun-Ping Wang, and Cang-Yuan Guo, "Modified Booth multipliers with a Regular Partial Product Array," IEEE Transactions on circuits and systems-II, vol 56, No 5, May 2009.
[3] Li-Rong Wang, Shyh-Jye Jou and Chung-Len Lee, "A well-structured Modified Booth Multiplier Design" 978-1-4244-1617-2/08/$25.00 c2008 IEEE.
[4] Soojin Kim and Kyeongsoon Cho "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges" World Academy of Science, Engineering andTechnology2010.
[5] Magnus Sjalander and Per Larson-Edefors. "The Case for HPM-Based Baugh-Wooley Multipliers," Chalmers University of Technology, Sweden, March 2008.
[6] Z Haung and M D Ercegovac, "High performance Low Power left to right array multiplier design" IEEE trans.Computer, vol 54 no3, page 272-283 Mar 2005.
[7] Hsing-Chung Liang and Pao-Hsin Huang, "Testing Transition Delay Faults in Modified Booth Multipliers by Using C-testable and SIC Patterns"IEEE2007, 1-4244-1272-2/07.
[8] Aswathy Sudhakar, and D. Gokila, "Run-Time Reconfigurable Pipelined Modified Baugh-Wooley Multipliers," Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 3 Number 2 (2010) pp. 223–235.
[9] Myoung-Cheol Shin, Se-Hyeon Kang, and In-Cheol Park, "An Area-Efficient Iterative Modified-Booth Multiplier Based on Self-Timed Clocking," Industry, and Energy through the project System IC 2010,and by IC Design Education Center (IDEC).
[10] Leandro Z. Pieper, Eduardo A. C. da Costa, Sergio J. M. de Almeida,"Efficient Dedicated Multiplication Blocks for2´s Complement Radix-2m Array Multipliers," JOURNAL OF COMPUTERS, VOL. 5, NO. 10, OCTOBER 2010.