# Design and Comparison of High Speed Radix-8 and Radix-16 Booth's Multipliers

Ila Chaudhary
Assistant Professor
Department of ECE, FET,
MRIIRS, Faridabad

Deepika Kularia
Student
Department of ECE, FET,
MRIIRS, Faridabad

Romika Choudhary
Assistant Professor
Department of ECE, FET,
MRIIRS, Faridabad

Gagandeep Kaur
Assistant Professor
Department of ECE, FET, MRIIRS, Faridabad

Ashish Vats
Assistant Professor
Department of ECE, FET, MRIIRS, Faridabad

## ABSTRACT
Multiplier is one of the hardware block which generally occupies a significant chip area and is required to be minimized which will be fruitful to number of applications in which multiplier blocks constitute an important unit such as digital signal processing (DSP) systems or computational techniques. Battery operated systems require low power devices to be implemented which can be minimized if the hardware required for the device is reduced logically. This paper focuses the DSP applications in which multiplier is significantly used and proposes a technique that helps in reducing the hardware as well as delay leading to the rise in performance of the system thus helping in increasing the operation frequency by a significant value. A 16-bit multiplier has been designed using a radix-8 and radix-16 Booth's multiplication that reduces number of partial products.

## General Terms
Algorithm, Multiplier, DSP Application, Xilinx

## Keywords
Radix-8 Booth's multiplier, Radix-16 Booth's multiplier, Partial products

## 1. INTRODUCTION
In digital design combinational logic implemented for computing the multiplication of two binary inputs leads to a large number of gate count which occupies a large chip area on the digital system. Moreover, it leads to large combinational delays in various digital systems such as Multiply and Accumulate(MAC) unit, computation Intensive Arithmetic Functions(CIAF) and is also being used in many Digital Signal Processing (DSP) applications such as Discrete Fast Fourier Transforms, Fast Fourier Transform(FFT), IIR and FIR filters, windowing techniques etc. Also, it is used in high end processors in order to reduce the computation complexity by using it in arithmetic and logic unit. Several multipliers such as: Array, Wallace tree, Vedic multiplier are available but Booth's multiplier has its own significance and advantages to be used in DSP applications[1].

In Array multiplier makes use of adding and shifting algorithm. In this algorithm a partial product is generated by the multiplication of multiplicand with each bit of the multiplier. After every multiplication partial product thus generated is shifted according to its bit order and then all the partial products are added to obtain the final product. Array multiplier is known for its simple, scalable and repetitive structure but occupies a large area as well as has significant combinational delay which many times contribute to the critical path delay.

Performance of the system in terms of speed can be enhanced by using Wallace tree multiplier instead of Array multiplier in which adders to add partial products are arranged in a tree like structure which reduces the combinational delay as well as the number of adder cells needed leading to the reduction in hardware area required. The propagation delay in the multiplier design comprising Wallace tree structure is equal to O (log3/2 (N)). Comparing with carry-save structure Wallace tree is quicker for higher word lengths, however, at the same time it shows unpredictable behavior, which leads to complex physical design implementation.

Madrid et. al. have compared Radix, Radix, 4, Radix 8, Radix 16 and Radix 32 multipliers and have proved Radix-4 to be the best multiplier in terms of the performance. However, it is not evident that whether the performance improves in terms of delay or area[2].

Minu Thomas published the simulation results of Radix-8 Booth Encoder Multiplier. In this paper, combinations of carry save adders and carry look ahead adders have been used for the addition of partial products in order to improve the performance of the multiplier for signed and unsigned numbers. Same multiplier circuitry has been used in the design for the multiplication of signed numbers as that for the unsigned numbers which helped in reducing the hardware requirement[3].

Swapna et al. have designed the methodology of 8-bit Radix-2 booth's multiplier in reversible mode. The multiplier designed can perform the multiplication of both signed and unsigned numbers which they have implemented without using any feedback as it is strictly prohibited in designs implemented by reversible logics[4].

## 2. BOOTH's MULTIPLIER
Booth's multiplication is meant for multiplying two's complement representation of signed binary numbers[5]. The algorithm is named to the credit of Andrew Donald Booth who devised it in 1950. Desk calculators were used for the computation and were found to be faster at shifting rather than adding and their speed was increased by Booth's algorithm. It has become a matter of interest for the applications making use of high end processors.

# 3. BOOTH's ALRORITHM

In order to multiply binary numbers,Booth's algorithm converts the multiplier Y in two's complement form and implicitly appends a bit $y_{-1} = 0$ below the least significant bit. It examines the pair of bits $y_i$ and $y_{i-1}$by executing the iterations starting from i = 0 to i = N-1. Following execution takes place on comparison:

1. The product in accumulator P is left unchanged if the compared bits are equal.
2. The multiplicand times $2^i$ is added to *P when* $y_i = 0$ and $y_{i-1} = 1$
3. The multiplicand times $2^i$ is subtracted from *P when* $y_i = 1$ and $y_{i-1} = 0$.

Finally a signed product P is obtained [6][7][8].

Typically, similar to the multiplier both the multiplicand and product are also in two's complement representation. However, the algorithm can be generalized to any number system which can supports addition and subtraction.

The ordering of the iterations is from LSB to MSB starting from i = 0. Accumulator P is shifted to one right for multiplication by 2 and LSB is shifted out in the case. Subsequent computation of addition and subtraction is then executed on the resulting N bits of P.

The calculation is regularly depicted as changing over series of 1s in the multiplier to a high-order +1 and a low-order – 1 at the closures of the string. At the point when a string goes through the MSB, there is no high-order +1, and the net impact is elucidation as a negative of the appropriate value[9][10].

$M * R = M * \{(S_{n-1} * 2^{n-1}) + (S_{n-2} * 2^{n-2}) \dots (S_2 * 2^2) + (S_1 * 2^1) + (S_0 * 2^0)\}$……..(1)

Where,

$p_{n-1} = S_{n-1} * 2^{n-1}$, $p_{n-2} = S_{n-2} * 2^{n-2}$.....$p_1 = S_1 * 2^1$, $p_0 = S_0 * 2^0$

So,

$M * R = pp_{n-1} * 2^{n-1} + pp_{n-2} * 2^{n-2}$ .....+ $pp_1 * 2^1 + pp_0 * 2^0$…….(2)

Where $ppn_{n-1} = (M * p_{n-1})$, $pp_{n-2} = (M * p_{n-2})$ .....$pp_1 = (M * p_1)$, $pp_0 = (M * p_0)$ are called partial products.

Add these 'n' partial products as shown in the equation below to get final product.

$M * R = pp_{n-1} * 2^{n-1} + pp_{n-2} * 2^{n-2}$..... + $pp_1 * 2^1 + pp_0 * 2^0$..... (3)

# 4. RADIX-8 BOOTH'S MULTIPLIER

Performance of the radix multiplier can be enhanced by introducing parallelism which results in reducing the number of calculation stages.

Radix-8 means: $8 = 2^3 = (1000)_2$

Radix-8 uses 4-bit

So, a group of 4-bit binary number is taken.

For a group of 4-bits the Signed Multiplier Digit is specified in Table. 1 which defines Radix-8 Booth's recoding technique for all possible combinations in the binary input where M is the Multiplier.

**Table 1: Radix-8 Booth's recoding technique**

| Multiplier bits | Signed Multiplier Digit |
|---|---|
| 0000 | 0 |
| 0001 | +1* M |
| 0010 | +1* M |
| 0011 | +2* M |
| 0100 | +2* M |
| 0101 | +3* M |
| 0110 | +3* M |
| 0111 | +4* M |
| 1000 | -4* M |
| 1001 | -3* M |
| 1010 | -3* M |
| 1011 | -2* M |
| 1100 | -2* M |
| 1101 | -1* M |
| 1110 | -1* M |
| 1111 | 0 |

There will be four partial products. Radix-8 Booth's Multiplier Sign Extension Trick defines the partial products as given below:

[Partial Product 1]

[Partial Product 2]0 0 0

[Partial Product 3] 0 0 0 0 0 0

[Partial Product 4] 0 0 0 0 0 0 0 0 0

# 5. RADIX-16 BOOTH'S MULTIPLIER

The technique of Radix-16 Booth's multiplication is explained further:

Radix-16 means: $16 = 2^4 = (10000)_2$

Radix-16 uses 5-bit

So, a group of 5-bitsis taken in the input binary number. Signed multiplier digit for the group is defined in Table 2 as per the Booth's recoding technique for every binary combination where M is the multiplier.

**Table 2:Radix-16 Booth's recoding strategy**

| Multiplier bits | Signed Multiplier Digit |
|---|---|

| | |
|---|---|
| 00000 | 0 |
| 00001 | +1* M |
| 00010 | +1* M |
| 00011 | +2* M |
| 00100 | +2* M |
| 00101 | +3* M |
| 00110 | +3* M |
| 00111 | +4* M |
| 01000 | +4* M |
| 01001 | +5* M |
| 01010 | +5* M |
| 01011 | +6* M |
| 01100 | +6* M |
| 01101 | +7* M |
| 01110 | +7* M |
| 01111 | +8* M |
| 10000 | -8* M |
| 10001 | -7* M |
| 10010 | -7* M |
| 10011 | -6* M |
| 10100 | -6* M |
| 10101 | -5* M |
| 10110 | -5* M |
| 10111 | -4* M |
| 11000 | -4* M |
| 11001 | -3* M |
| 11010 | -3* M |
| 11011 | -2* M |
| 11100 | -2* M |
| 11101 | -1* M |
| 11110 | -1* M |
| 11111 | 0 |

There will be five partial products as per the Booth's multiplier sign extension trick as explained further:

Radix-16 Booth's Multiplier Sign Extension Tricks.

[Partial Product 1]

[Partial Product 2]0000

[Partial Product 3]00000000

[Partial Product 4] 000000000000

[Partial Product 5] 0000000000000000

## 6. RESULTS AND STIMULATION

Fig. 1 and Fig. 2 show the waveform results for Radix-16 and Radix-8 multiplier for 16-bit numbers respictively. In Table 3 device utilization summary has been shown.

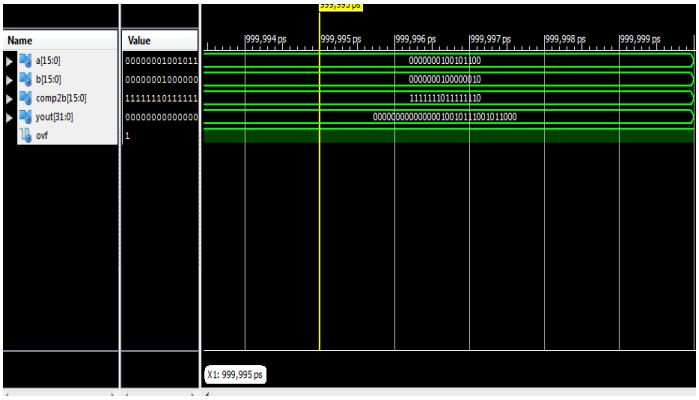


**Fig.1: Output stimulation of Radix-16 multiplier**

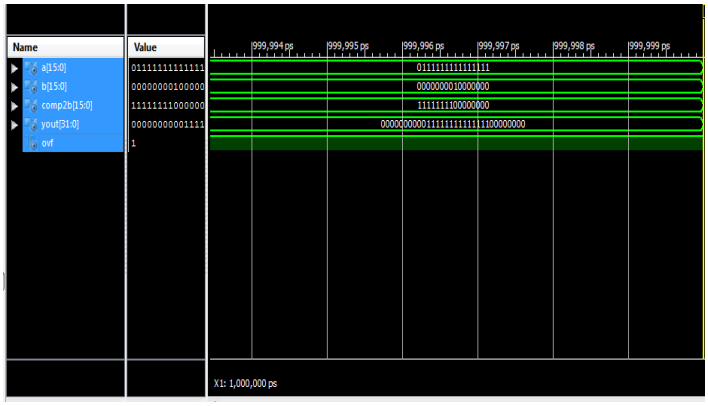


**Fig. 2: Output stimulation of Radix-8 multiplier**

The VHDL code of 16×16 bit Radix-16 multiplier was synthesized using Xilinx ISE 14.4 on virtex4 family device XC4VLX25 and the results are shown in Fig. 4. Comparison of area and delay is shown in table1. In which Radix-8 and Radix-16 multiplier 16×16 bit is stimulated on xc3s400-5tql44 of SPARTAN 3 and rest on xc4vlx25-12ff676 of vitex 4.

**Table 3: Device utilization summary**

| Logic Utilization | Used | Available | Utilization |
|---|---|---|---|
| Number of 4 Input LUTs | 1,188 | 21,504 | 5% |
| Number of occupied slices | 620 | 10,752 | 5% |
| Number of slices containing only related logic | 620 | 620 | 100% |
| Number of slices containing unrelated logic | 0 | 620 | 0% |
| Total Number of 4 input LUTs | 1,190 | 21,504 | 5% |
| Number as a logic | 1,188 | | |

| | | | |
|---|---|---|---|
| Number used as a route-tru | 2 | | |
| Number of bonded I/Os | 81 | 440 | 18% |
| Average fan-out of Non-clock Nets | 3.90 | | |

Synthesis report result for the timing details is shown in Fig. 3

```
Timing Summary:
---------------
Speed Grade: -12

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 25.004ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 1770310 / 33
-------------------------------------------------------------------------
Delay:               25.004ns (Levels of Logic = 35)
  Source:            a<3> (PAD)
  Destination:       ovf (PAD)

  Data Path: a<3> to ovf
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             55   0.754   1.000  a_3_IBUF (a_3_IBUF)
     LUT4:I1->O            16   0.147   0.727  y1_cmp_eq00261 (y1_cmp_eq0026)
```

**Fig. 3: Timing Details**

Fig.4 shows the RTL schematic of the design synthesized for Radix 8 multiplier**.**

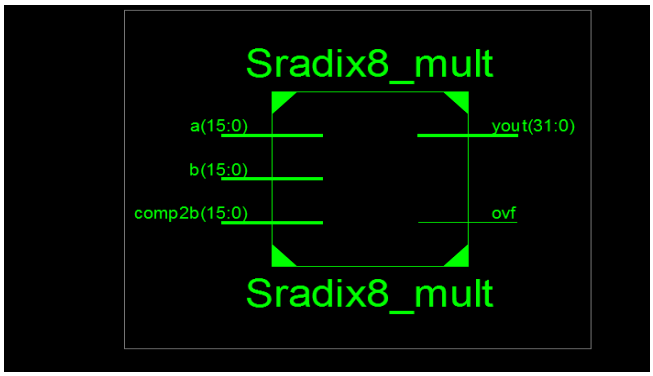


**Fig.4: RTL schematic of Radix-8 multiplier**

Radix-8 and Radix-16 multipliers designs have been compared in terms of their area and delay in Table 4.

**Table 4: Comparison Table of designed architectures**

| Design | Delay | Area | |
|---|---|---|---|
| | | No. of 4 input LUTs | No. of occupied slices |
| Radix-8(16x16) | 25.179ns | 773/ 21504 | 442/ 10752 |
| Radix-16(16x16) | 25.004ns | 1188/ 21504 | 620/ 10752 |

Fig. 5a and 5b show the comparison of Delay and the area occupied by both Radix 8- Radix -16 multipliers.

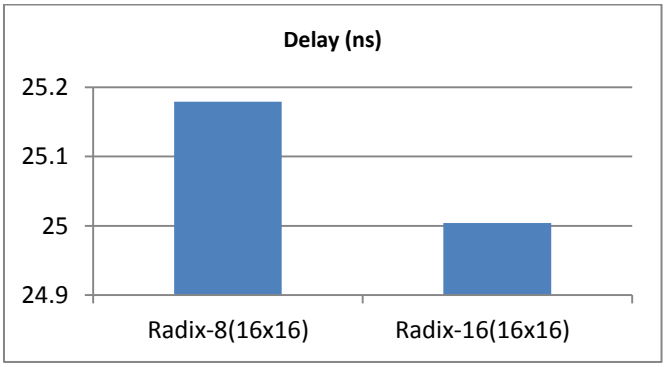


**Fig 5a: Delay in both Radix-8 and Radix-16 Multipliers**

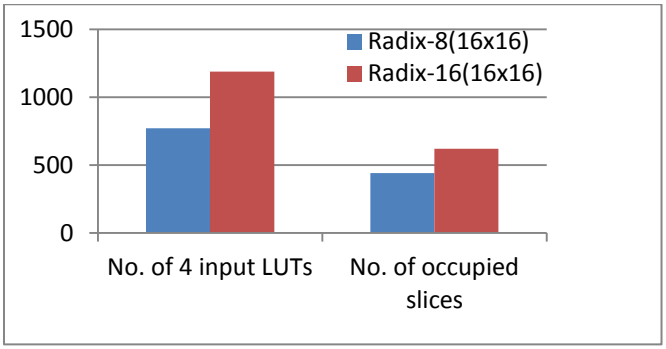


**Fig. 5b: Area Utilization in Radix-8 and Radix-16 Multipliers**

## 7. CONCLUSION

In this paper, radix-8 and radix-16 booth's multiplication algorithms were carried out for the DSP applications. From table 3 it can be concluded that radix-16 takes less delay but more area than that of radix-8 booth's multiplication. However, the distinction is not always appreciable in phrases of delay, because of the rise in computational complexity of radix -16 technique owing to large number of partial products. If we apply radix-8 technique for designing high end processors such as 32-bit or 64-bit processor, then it will cost in terms of large on chip area due to larger number of groups and multiplicand. Therefore, the selection of booth's algorithm depends upon the specification requirements of the particular application.

It is found that the results vary to a large extent by making fewer changes in the methodology, technique or hardware logics. There is a continuous research on different Radix multipliers[11]. Also, different adders being used can also impact the results to a large extent depending on the scenario and the application. So, there is a scope of analysis to be carried out for the kind of architecture being used for the multiplication depending on the number of bits in the numbers being multiplied or whether they are signed or unsigned numbers.

# 8. REFERENCES

[1] A B. Pawar , "Radix-2 Vs Radix-4 High Speed Multiplier", International Journal ofAdvanced Research in Computer Science and Software Engineering, Volume 5, Issue 3, pp. 329-333, March 2015.

[2] Philip E. Madrid and Brian Millar, "Modified Booth Algorithm for High Radix Multiplication",IEEE Transactions on Very Large Scale Integration (VLSI) Systems Volume.1, Issue 2, pp. 164 – 167, August 2002.

[3] Minu Thomas, "Design and Simulation of Radix-8 Booth Encoder Multiplier for Signed and Unsigned Numbers", International Journal for Innovative Research in Science & Technology| Vol. 1, Issue 1, pp 1-10June 2014.

[4] K.Swapna, A.krishna Mohan, "Area Optimized Radix-2 8-Bit Reversible Booth Multiplier", Int. Journal of Engineering Research and Application, Vol. 7, Issue 10, pp.65-70, ( Part -1) October 2017.

[5] Kelly Liew Suet Swee, Lo Hai Hiung, "Performance comparison review of Radix-based multiplier designs", 4th International Conference on Intelligent and Advanced Systems, Volume 2, pp 854- 859 12-14th June 2012.

[6] Chandrashekhar T. Kukade,"A Novel Parallel Multiplier for 2's Complement Numbers Using Booth's Recoding Algorithm", IEEE,International Conference on Electronic Systems, Signal Processing and Computing Technologies, Volume 2, Issue 8, pp.93 – 98, 9-11 Jan. 2014.

[7] CHEN Ping-hua, ZHAO Juan. "High-speed Parallel 32×32-b Multiplier Using a Radix-16 Booth Encoder", IEEE, Third International Symposium on Intelligent Information Technology Application Workshops, Volume 3, Issue 4, pp. 406-409, Nov. 2009.

[8] Laya Surendran E K, Rony Antony P, "Implementation of fast multiplier using modified Radix-4 Booth Algorithim with redundant binary adder for low energy applications", First International Conference on Computational Systems and Communications, Volume 1, Issue 2, pp.266-271, 17-18 Dec 2014.

[9] Na Tang, "A High-Performance 32-bit Parallel Multiplier Using Modified Booth's Algorithm and Sign-Deduction Algorithm", IEEE,ASIC,Volume.2 ,pp. 1281 – 1284,Oct 2003.

[10] Razaidi Hussin, Ali Yeon Md. Shakaff, "An Efficient Modified Booth Multiplier Architecture", IEEE, International Conference on Electronic Design, Volume 1, Issue 6, pp.271-276, December 2008.

[11] Kajal B. Bobade, Prof. V. G. Roy, Prof. S. Kuntawar, "A Review On Fast Radix-10 Multiplication Using Binary Input And Convert Into Decimal Codes", International Journal of Science, Engineering and Technology Research, Volume 06, Issue 05, pp 795-797, May 2017.