# REALIZATION OF DECIMAL MULTIPLICATION USINGRADIX-16 MODIFIED BOOTH ENCODINGALGORITHM

**N. JAGADEESH CHANDRA MURTHY, E.V. NARYANA**

*VLSI AND EMBEDDED SYSTEMS, JNTUK*
*ASST. PROFESSOR, JNTUK*

**Abstract - This paper presents the concept of decimal multiplication using modified booth algorithm. It is one of the most important decimal arithmetic operations which have a growing demand in the area of commercial, financial, and scientific computing. In this paper, Multiplier is one of the key hardware blocks in most digital and high performance systems such as FIR filters, digital signal processors and microprocessors etc. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following- high speed, low power consumption, regularity of layout and hence less area or even combination of them in multiplier. Thus making them suitable for various high speed, low power, and compact VLSI implementations. However area and speed are two conflicting constraints. So improving speed results always in larger areas. So here we try to find out the best trade off solution among the both of them. Generally as we know multiplication goes in two basic steps. Partial product and then addition. Then we turned to Booths Multiplier and designed Radix-16 modified booth multiplier and analyzed the performance of multiplier.**

*Index Terms*— **Radix-16 multiplier, Modified recoder(MBR), Booth encoder(BE),VLSI design.**

## I. Introduction

Multiplication (Series of repeated additions) is an essential function in basic arithmetic operations, especially in signal processing applications, includes graphics and computation system (Mead, C. and Conway, L., 1988). The factorization of a large number checks whether a number is prime. It depends on multiplication. Multiplicand is the number that is added and the result of it is the product. The modified stand-alone multiplier consists of a modified recorder (MBR). MBR has two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The operation of BE is to decode the multiplier signal, and the output is used by BS to produce the partial product.

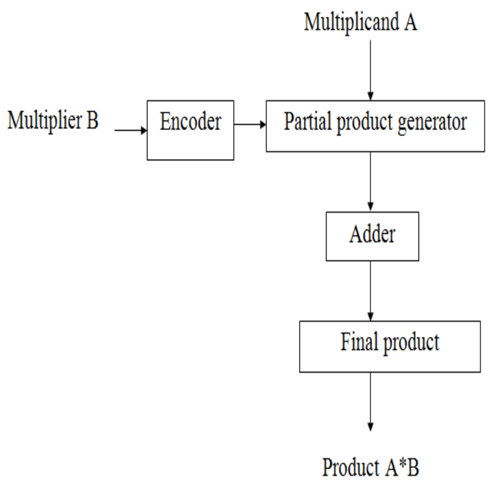


Fig1: booth algorithm

## II. Modified Booth Algorithm

The multiplier architecture consists of two architectures, i.e., Modified Booth. By the study of different multiplier architectures, we find that Modified Booth increases the speed because it reduces the partial products by half. Also, the delay in the multiplier can be reduced by using Wallace tree. The energy consumption of the Wallace Tree multiplier is also lower than the Booth and the array. The characteristics of the two multipliers can be combined to produce a high-speed and low-power multiplier.

The modified stand-alone multiplier consists of a modified recorder (MBR). MBR has two parts, i.e., Booth Encoder (BE) and Booth Selector (BS). The operation of BE is to decode the multiplier signal, and the output is used by BS to produce the partial product. Then, the partial products are added to the Wallace tree adders, similar to the carry-save-adder approach. The last transfer and sum output line are added by a carry look- ahead adder, the carry being stretched to the left by positioning.

Booth multiplication algorithm consists of three major steps as shown in the structure of booth algorithm figure that includes generation of partial product called as recoding, reducing the partial product in two rows, and addition that gives final product. For a better understanding of modified booth algorithm & for multiplication, we must know about each block of booth algorithm for multiplication process.

**III. Modified Booth Algorithm Encoder**

This modified booth multiplier is used to perform high-speed multiplications using modified booth algorithm. This modified booth multiplier's computation time and the logarithm of the word length of operands are proportional to each other. We can reduce half the number of partial product. Radix-4 booth algorithm used here increases the speed of multiplier and reduces the area of multiplier circuit. In this algorithm, every second column is taken and multiplied by 0 or +1 or +2 or -1 or -2 instead of multiplying with 0 or 1 after shifting and adding of every column of the booth multiplier. Thus, half of the partial product can be reduced using this booth algorithm. Based on the multiplier bits, the process of encoding the multiplicand is performed by radix-4 booth encoder.

Multiplier: $B=-bn-12n-1+\sum n-2i=0bi2i$

- Introduce new variables: $b\hat{}i=-bi+bi-1$ for $i=0\ldots n-1$ (assume $b-1=0$).
- Compute $P=\sum n-1i=0(b\hat{}i\times 2i\times A)$
- Although this looks very similar to the normal product, note that any time $bi=bi-1$ there is no addition to be performed as the partial product will be zero. In the case of serial addition, these steps can be skipped, thus saving computation

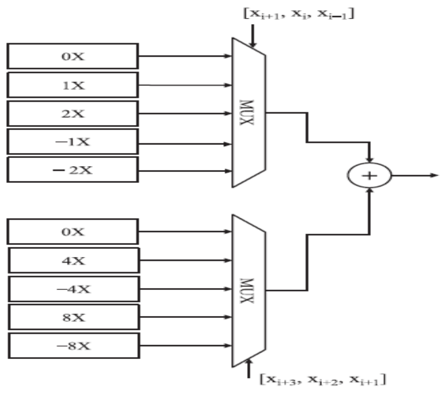


Figure 2:Radix16 modified booth encoder

To reduce the number of partial products added while multiplying the multiplicand higher radix Booth Encoding algorithm is one of the most well known techniques used .Radix 16 Booth algorithm which scan strings of five bits with the algorithm given below:

(1) Extend the sign bit 1 position if necessary to ensure that n is even.

(2) Append a 0 to the right of the LSB of the multiplier.

(3) According to the value of each vector, each Partial Product will be 0,y,+2y,+3y,+4y,+5y,+6y,+7y,+8y, -8Y, -7y, - 6y, - 5y, - 4y, -3y, - 2Y, - Y. The multiplication of y is done by shifting y by one bit to the left. Thus, in any case, in designing n bit parallel multipliers, only n/4 partial products are generated

**IV. MODIFIED BOOTH PROCEDURE**

Multiplication is one of the most important and basic arithmetic operation that constitute programs. In fact 8.72% of all instructions in typical scientific programs are based on multiplication operation [2]. Many multipliers have been proposed in the past with consideration of small area, low power and high performance. Multiplication is achieved by the addition of a certain number of partial products rows. Each partial product row is generated by multiply the multiplier bit one by one to multiplicand. In a simple multiplier, the generated partial products rows are equal to the number of bits in multiplier. For example, in 8×8 bit multiplication, it will produce 8 partial product rows. It will take more adders and more time. To improve the performance of the multiplier, Booth multiplier is mostly used multiplier. The number of partial products rows that must be added to give the multiplications result can be reduced by using Booth decoding. In Booth multiplier, the numbers of reduced partial products rows are depend on the grouping done at multiplier bits. These groups of multiplier perform the selected operation on multiplicand. In booth multiplier grouping is done by 2 bits, 3 bits, 4 bits and so on. Higher order booth decoding reduces the number of partial product rows by a greater by decoding larger groups of multiplier bits. This multiplication process is completed in 3 steps. First step: multiplier bits are divided in groups then these groups are fed to decoder at where it will indicate that which operation is to perform on multiplicand. Second step: here indicated operation performs on the multiplicand and it will generate the partial products. Third step: Now generated partial products are adding with adders.

For requirements of smaller area occupation and faster operation. This is suitable for 2's complementary and signed number multiplication.

- Pad the LSB with one zero.

- Pad the MSB with 2 zeros if n is even and 1 zero if n is odd.
- Divide the multiplier into overlapping groups of n-bits based on radix using
- Determine partial product scale factor from modified booth encoding table.
- Compute the Multiplicand Multiples.
- Sum the Partial Products obtained

**ALGORITHM STEPS**

- Append '0' to multiplier
- Multiplier recoding
- Booth encoding
- Partial product generation

## V. RESULTS

After design entry and optimal simulation, you run synthesis. During this step Verilog Language designs become net list files that are accepted as input to the implementation step. The counters Synthesis & simulation results are:

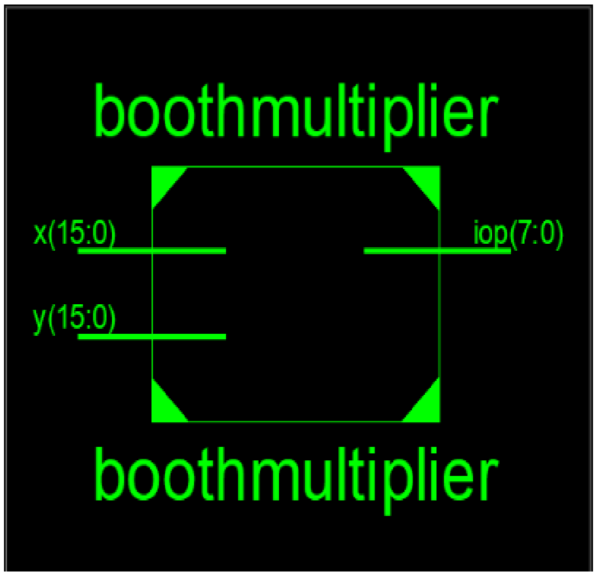


Figure 3: RTL schematic of modified radix 16 booth multiplier

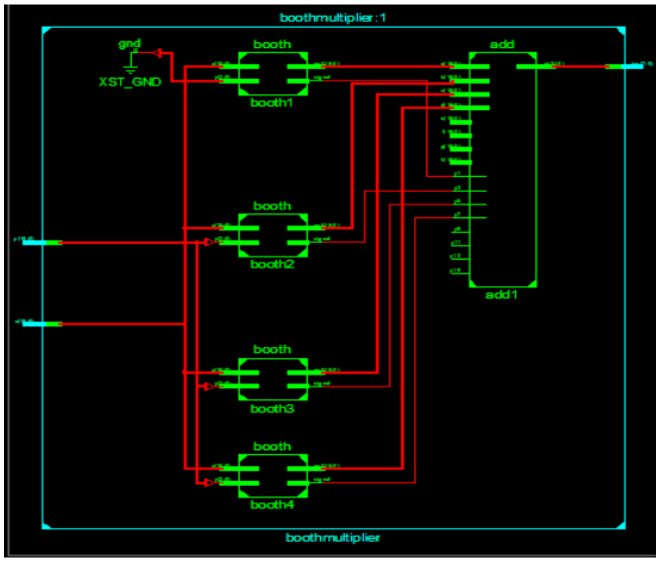
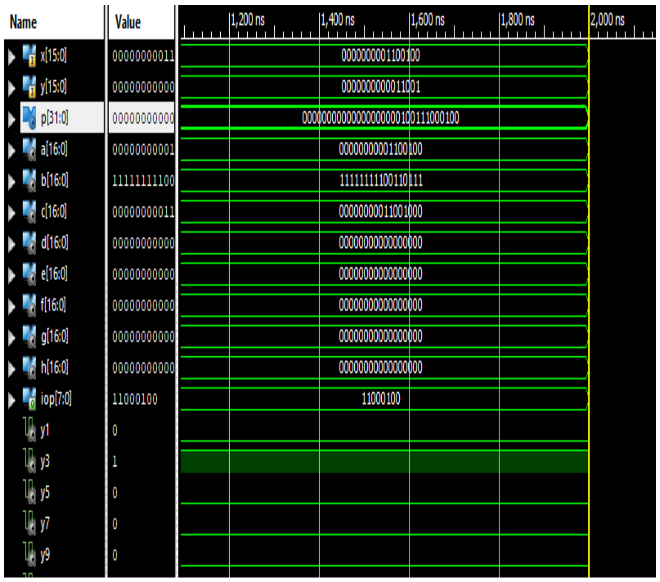


Figure 4. Technological view of Decimal multiplication



```
Name      Value          1,200 ns   1,400 ns   1,600 ns   1,800 ns   2,000 ns
x[15:0]   00000000011    0000000001100100
y[15:0]   00000000000    0000000000011001
p[31:0]   00000000000    00000000000000000000100111000100
a[16:0]   00000000001    00000000001100100
b[16:0]   11111111100    11111111100110111
c[16:0]   00000000011    00000000011001000
d[16:0]   00000000000    00000000000000000
e[16:0]   00000000000    00000000000000000
f[16:0]   00000000000    00000000000000000
g[16:0]   00000000000    00000000000000000
h[16:0]   00000000000    00000000000000000
iop[7:0]  11000100       11000100
y1        0
y3        1
y5        0
y7        0
y9        0
```

Figure.5: Simulation Result of modified radix 16 booth multiplier

## VI. CONCULSION

In Multipliers we studied different Multipliers starting from Array Multiplier to Wallace Tree, Booth Multipliers, both Radix-4 and Radix-16.We found that parallel multipliers are much better than the serial multipliers due to less area consumption and hence the less power consumption. Comparing Radix-4 and Radix-16 booth multipliers we found that radix-16 consumes less power than radix-4, because radix-16 uses almost half number of iterations than radix-4.We saw Wallace tree having nearly same delay as of radix-16 multipliers where as consuming a little more power than the former. After all this then we tried to improve power efficiency of circuits. Hence we went for studying different recoding schemes along with their Partial Product generators and study time and power required by them in a multiplication process. After studying them we went to modify one of the recoding schemes to find a proper combination of recoder and PP generator such that we will have simplest PP generator as these take maximum area in a cell area and then take care of zero handling as it handles most of the switching activities. Hence we ended up creating a better recoding scheme.

## VII. REFERENCES

1. A. Aswal, M.G. Perumal, and G.N.S. Prasanna, "On basis financial decimal operations on binary machines," IEEE trans. Comput., vol. 61, no. 8, pp. 1084-1096, Aug. 2012.

2. M.F. Cowlishaw, E.M. Schwarz, R.M. Smith, and C.F. Webb, "A decimal floating-point specification," in Proc. 15th IEEE symp. Comput.Arithmatic, June 2001, pp. 147-154.

3. M.F. Cowlishaw, "Decimal floating- point: Algorithm for computers," in Proc. 16th IEEE symp. Comput.Arithmatic, Jully 2003.

4. S. Carlough and E. Schwarz, "Power6 decimal divide," in Proc. 18th IEEE Symp.

5. S. Carlough, S. Mueller, A. Collura, and M. Kroener, "The IBM zEnterprise-196 decimal floating point accelerator".

6. L. Dadda, "Multioperand parallel decimal adder: A mixed binary and BCD approach," IEEE Trans. Comput., vol. 56, no.10, pp. 1320-1328, oct. 2007.

7. L. Dadda and A. Nannarelli, "A variant of a Radix-10 combinational multiplier," in proc. IEEE Int. Symp.Circuitssyst, May 2008, pp. 3370-3373.

8. L. Han and S. Ko, "High speed parallel decimal multiplication with redundant internal encodings," IEEE Trans. Comput., vol. 62, no. 5, pp. 956–968, May 2013