Journal of Science Technology and Research (JSTAR)

Comparative Analysis of 16-Bit Booth Multiplier Using Radix-4 and Radix-8 Encoding Technique

[1]Ch.Pallavi, [2]C.Rajani

[1, 2]Assistant Professor, Department of ECE, SV Engineering College, Andhra Pradesh, India.

[1]pallavi.ch@svscolleges.edu.in, [2]rajani.chinta@svscolleges.edu.in

Abstract: Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers for high speed, low power consumption, regularity of layout and hence less area. Booth Multiplier can be used for the operation of both signed and unsigned numbers. The proposed radix-4 and radix-8 booth multiplier compare in terms of the number of partial products, delay and frequency. The numbers of partial products are reduced to n/2 in radix-4. We can reduce the number of partial products even further to n/3 by using a higher radix- 8 in the multiplier encoding, thereby obtaining a simpler CSA tree. The CSA (carry save adder) tree and the final CLA (carry look ahead adder) used to speed up the multiplier operation. Since signed and unsigned multiplication operation is performed by the same multiplier unit. So, the required hardware and chip area reduces and in turn reduces power dissipation and complexity. Power dissipation is recognized as a critical parameter in modern VLSI design field.

Key words: carry save adder (CSA), carry look ahead adder (CLA), Booth Multiplier, radix- 4, radix-8.

Introduction:

Multiplication is a most commonly used operation in many computing systems. In fact, multiplication is nothing but addition since, multiplicand adds to itself multiplier number of times gives the multiplication value between multiplier and multiplicand. But considering the fact that this kind of implementation really takes huge hardware resources and the circuit operates at utterly low speed. In order to address this so many ideas have been presented so far for the last three decades.

Corresponding Author: Ch.Pallavi, Asst. Professor,
Department of ECE, SV Engineering Colleges,
Andhra Pradesh, India.
Mail: pallavi.ch@svcolleges.edu.in

Each one is aimed at particular improvement according to the requirement. One may be aimed at high clock speeds and another maybe aimed for low power or less area occupation. Either way ultimate job is to come up with an efficient architecture which can address three constraints of VLSI speed, area, and power [1]. Among these three speeds is the one which requires special attention. The Multiplication operation involves two steps one is producing partial products and adding these partial products. Thus, the speed of a multiplier hardly depends on how fast generate the partial products and how fast we can add them together. If the numbers of partial products to be generated are of less than it is indirectly means that we have achieved the speed in generating partial products.

Booth's algorithms are meant for this only. To speed up the addition among the partial products we need fast adder architectures. Since the multipliers have a significant impact on the performance of the entire system, many high-performance algorithms and architectures have been proposed. The very high speed and dedicated multipliers are used in pipeline and vector computers [2]. The high-speed Booth multipliers and pipelined Booth multipliers are used for digital signal processing (DSP) applications such as for multimedia and communication systems.

MULTIPLIER ARCHITECTURE

A circuit that multiplies two unsigned n bit binary numbers uses a 2-dimensional array of identical sub circuits. Each of which contains a full adder and an and gate. For large number of bits this approach may not be appropriate because of the large number of gates needed. Another approach is to use shift register in combination with an adder to implement the traditional method of multiplication [3].

TYPES OF MULIPLIERS

Serial Multiplier, Serial/Parallel Multiplier, Array Multiplier, Baugh Wooley Multiplier, Booth Multiplier and Modified Booth Multiplier. There are several Multiplier Architectures which has come into existence over recent years. Multiplier is one of the key hardware blocks in Digital Signal Processors (DSP) and microprocessors. Multiplication operations are so considerable in-order to slow down the system operations as shown in figure 1 [4].

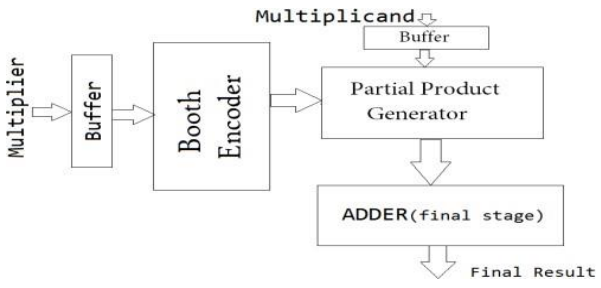


Fig 1. Multiplier Architecture

Multiplication process has four main steps

1. Booth Encoding 2. Partial product summation 3. Final addition 4. Accumulation

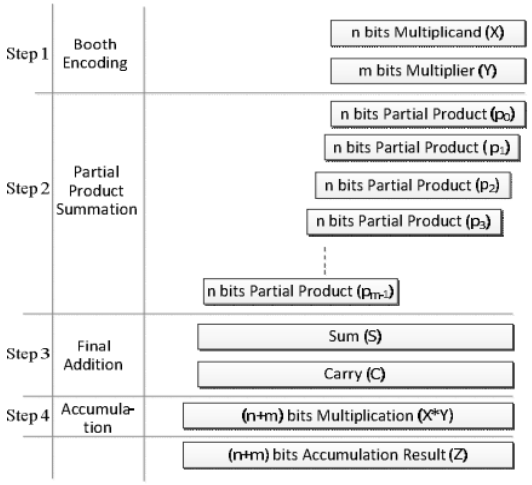


Fig 2. Basic arithmetic steps for multiplication

CHARACTERISTICS OF MULIPLIERS

Accuracy: - A good multiplier should give correct result.
Speed:-Multiplier should perform operation at high speed.
Area: - A multiplier should occupy a smaller number of slices and LUTs.
Power: - Multiplier should consume less power.

This paper is organized as follows section II explains Existing Radix-2 and Radix 4 Booth Multiplier. In section III describes the proposed Radix 8 Modified booth multiplier. In section IV describes the results and comparative analysis and finally section V describes the Conclusion and future scope.

EXISTING SYSTEM

BOOTH MULTIPLIER

Booth algorithm was first implemented by Andrew Donald Booth 1950. In normal Booth's algorithm is an efficient hardware implementation of a digital circuit that multiplies two binary numbers in two's complement notation. Booth multiplication is a fastest technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied [5], [10].

Booth multiplication algorithm gives a procedure for multiplying binary integers in signed -2's complement representation. In booth multiplication process arithmetic shift and circular shift operations are performed.

Table 1: Radix 2 Encoding Table

Yi	Yi-1	Partial Product
0	0	0*Multiplicand
0	1	1*Multiplicand
1	0	-1*Multiplicand
1	1	0*Multiplicand

In this algorithm, the Yi and Yi-1 bits of the multiplier are examined and then recoding is done as shown in table 1. Booth Recoding reduces the number of partial products which can reduce the hardware and improves the speed of the operation. Some considerable delay is seen during the generation of partial products. This radix-2 Booth recoding works well with serial multiplication which can tolerate variable latency operations with minimum number of serial additions [6]. In case of parallel multiplication, the worst case is seen. The worst case is a n-bit Multiplicand significantly requires n - no. of additions. Here the no. of partial products seen are also ' n '. The number of partial products in Radix-2 for n bit is 'n'.

MODIFIED BOOTH MULTIPLIER

It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. For the standard add-shift operation, each multiplier bit generates one multiple of the multiplicand to be added to the partial product as shown in figure 2. If the multiplier is very large, then a large number of multiplicands have to be added [7]. In this case the delay of multiplier is determined mainly by the number of additions to be performed.

RADIX 4 MODIFIED BOOTH MULTIPLIER

It is possible to reduce the number of partial products by half, by using the technique of radix 4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column and multiply by ±1, ±2, or 0, to obtain the same results [8]. Radix 4 booth encoder performs the process of encoding the multiplicand based on multiplier bits. It will compare 3 bits at a time with overlapping technique. Grouping starts from LSB and the first block only uses two bits of multiplier and assumes a zero the third bit. When forming the Second group, the first bit is considered as the MSB bit of first group and remaining two bits from multiplier. This repeats for next group formation also.

The Booth Radix-4 algorithm reduces the number of Partial products by half at minimum circuit's complexity. This Radix- 4 can also be called as Modified Booth Algorithm [9].

Booth recoding is carry free and completely performed parallel to have speed. The main bottleneck in the speed of multiplication is the addition of partial products. More the number of bits the multiplier/multiplicand is composed of, more are the number of partial products, longer is the delay in calculating the product.

The critical path of the multiplier depends upon the number of partial products. In radix-2 booth's algorithm, if we are multiplying 2 'n' bits number, we have 'n' partial products to add. Radix-4 booth's multiplication is an answer to reducing the number of partial products. Using Radix-4 booth's multiplier, the number of partial products are reduced to 'n/2' if we are multiplying two 'n' bits numbers, if 'n' is even number, or '(n+1)/2', if 'n' is an odd number. By reducing the number of partial products, one can effectively speed up the multiplier by a factor roughly equal to 2.

ALGORITHM

- ✓ The Least Significant Bit (LSB) should be padded with zero (Y-1 = 0).
- ✓ For Signed, the MSB must be padded with two zeros when n is even or else onezero. For Unsigned, the padding is not necessary when n is even.
- ✓ Grouping of Multiplier into 3-bits must be done which in turn are overlapping.
- ✓ Partial products are generated with the help of Booth's recoding table as stated above.
- ✓ On adding the partial products, the result can be found as shown in table 2.

Table 2. Radix 4 Encoding

Yi+1	Yi	Yi-1	Partial Products
0	0	0	0*Multiplicand
0	0	1	1*Multiplicand
0	1	0	1*Multiplicand
0	1	1	2*Multiplicand
1	0	0	-2*Multiplicand
1	0	1	-1*Multiplicand
1	1	0	-1*Multiplicand
1	1	1	0*Multiplicand

The Radix-4 Booth recoding works effectively for both Signed and Unsigned numbers. The number of partial products in Radix-4 for n bit is 'n/2'.Booth's algorithm can be implemented by repeatedly adding one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier,respectively; and let x and y represent the number of bits in m and r [11]. Determine the values of A and S, and the initial value of P.

PROPOSED SYSTEM

MODIFIED BOOTH MULIPLIER

High speed multipliers are fundamental elements in signal processing and arithmetic

based systems. The higher bit widths required of modem multipliers provide the opportunity to explore new architectures which would be impractical for smaller bit width multiplication. Architectures for circuit elements historically were designed to operate at maximum speed, notwithstanding the resulting power dissipation as shown in figure 3. Recently, greater emphasis has been placed on reducing the power dissipation of important circuit functions while maintaining these high speeds. Therefore, power dissipation as well as circuit speed should be considered at the architectural level.

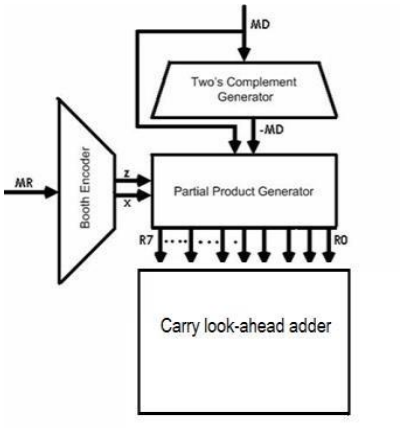


Fig 3. Block diagram of Modified Booth Multiplier

Multiplication is one of the four elementary mathematical operations of arithmetic with the others being addition, subtraction and division. The multiplication of whole numbers may be thought as a repeated addition; that is, the multiplication of two numbers is equivalent to adding as many copies of one of them, the multiplicand, as the value of the other one, the multiplier. Normally, the multiplier is written first and multiplicand second.

The multiplier takes in 2 16-bits operands the multiplier (MR) and the multiplicand (MD), then produces 32-bit multiplication result of the two as its output. The architecture comprises four parts: Complement Generator, Booth Encoder, Partial Product and Carry Look- ahead Adder. We adapt the simplest way to demonstrate the multiplier.

The modified-Booth algorithm is extensively used for high-speed multiplier circuits. Once, when array multipliers were used, the reduced number of generated partial products significantly improved multiplier performance. In designs based on reduction trees with logarithmic logic depth, however, the reduced number of partial products has a limited impact on overall performance.

The Continuous advances of the microelectronic technologies make better use of input energy, to encode the data more efficiently, to transmit the information faster and reliable, etc. In Particular, many of these technologies handle low power consumption to meet the requirements of various outboard applications. In these applications, a

multiplier is a fundamental arithmetic unit and used in a great extent in circuits.

RADIX-8 MODIFIED BOOTH MULTIPLIER

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 [1]. Thus it allows a time gain in the partial products summation as shown in figure 4. Modified Booth's is twice as fast as Booth's algorithm. Modified Booth

encoding algorithm is an efficient way to reduce the number of partial products by grouping consecutive bits in one of the two operands to form the signed multiples. The operand that is Booth encoded is called the multiplier and the other operand is called the multiplicand. Booth encoding table represents the number of ones and zeros present in the encoding.

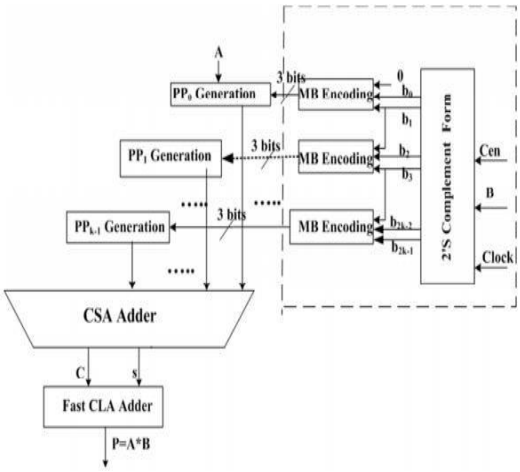


Fig 4. Modified Radix-8 Booth Multiplier

The output bits are calculated with the help of multiplicand which gives the information about sign bit, number of zeros and number of ones. For the design of high-performance modified radix 8 booth multiplier the following steps are applied.

Let the inputs be Multiplier (r) and Multiplicand (m).

- ✓ We take 2's compliment for Multiplicand (-m), and then the outcomes are given to modified booth encoding algorithm.
- ✓ Then 3 bits are produced.
- ✓ These bits are multiplied with Multiplier (r).
- ✓ Then we obtain the partial products & these given inputs to the carry save adder.
- ✓ These carry and sum are added using carry look head adder.
- ✓ Finally, the multiplier output is obtained.
- ✓ In this Radix-8 Booth's Algorithm the multiplier has been divided in to groups of four bits and these are overlapping groups [1].

The first group is formed by taking the first bit as zero, and the remaining three bits are the three LSB bits of Multiplier. When forming the Second group, the first is considered as the MSB bit of first group and remaining from multiplier. This repeats for next group formation also. Now, the Partial products are then generated by the help of three group bits with the Radix-8 recoding as shown in table 3.

Table 3. Radix 8 encoding

Multiplier Group Bits	operation
0000	0*Multiplicand
0001	+1*Multiplicand
0010	+2*Multiplicand
0011	+3*Multiplicand
0100	+4*Multiplicand
0101	+5*Multiplicand
0110	+6*Multiplicand
0111	+7*Multiplicand
1000	-7*Multiplicand
1001	-6*Multiplicand
1010	-5*Multiplicand
1011	-4*Multiplicand
1100	-3*Multiplicand
1101	-2*Multiplicand
1110	-1*Multiplicand
1111	0*Multiplicand

The number of partial products in Radix-8 for n bit is ‘n/3’. Booth's algorithm can be implemented by repeatedly adding one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r. Determine the values of A and S, and the initial value of P.

All of these numbers should have a length equal to (x + y + 1) A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros. S: Fill the most significant bits with the value of (−m) in two's complement notation. Fill the remaining (y + 1) bits with zeros. Determine the two least significant (rightmost) bits of P.

- If they are 0000, find the value of P.
- If they are 0001, find the value of P + A. Ignore any overflow.
- If they are 0010, find the value of P + A. Ignore any overflow.
- If they are 0011, find the value of P + 2A. Ignore any overflow.
- If they are 0100, find the value of P +2 A. Ignore any overflow.

- If they are 0101, find the value of P + 3 A. Ignore any overflow.
- If they are 0110, find the value of P + 3 A. Ignore any overflow.
- If they are 0111, find the value of P + 4 A. Ignore any overflow.
- If they are 1000, find the value of P + 4S. Ignore any overflow.
- If they are 1001, find the value of P + 3S. Ignore any overflow.
- If they are 1010, find the value of P + 3S. Ignore any overflow.
- If they are 1011, find the value of P + 2S. Ignore any overflow.
- If they are 1100, find the value of P + 2S. Ignore any overflow.
- If they are 1101, find the value of P + S. Ignore any overflow.
- If they are 1110, find the value of P + S . Ignore any overflow.
- If they are 1111, find the value of P .

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-unsigned s_u=0, it indicates unsigned number multiplication and when s_u=1, it indicates signed number multiplication.

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.[12] 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.

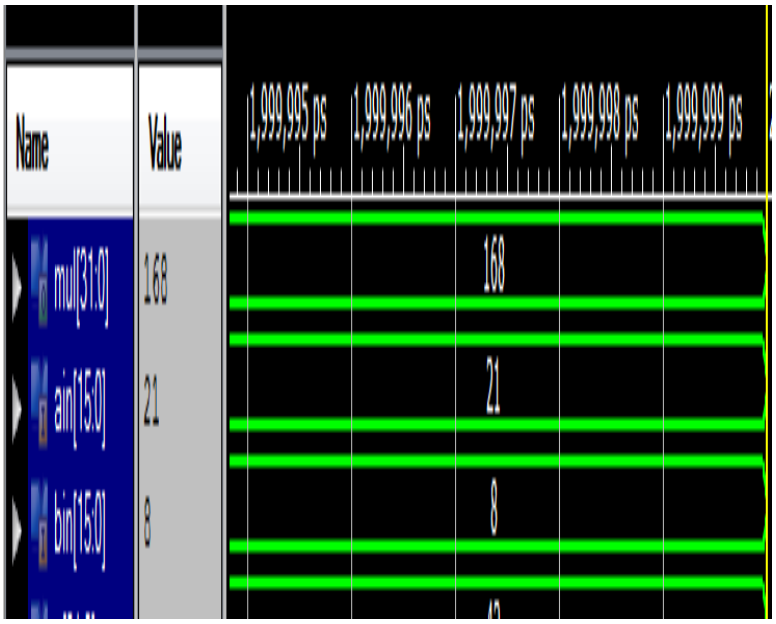
To generate 3a we have to add 2a and a, and similarly for -3a add -2a and –a. Here 3a and -3a are hard multipliers. The number of partial products are reduced to 'n/3' in radix 8 modified booth multiplier. So the delay in the simulation decreases and the speed of the performance of the operation increases. As the partial products are reduced the area utilization for multiplication is reduced.

Radix 8 booth encoding multiplier

[13], [14] uses 4-bit encoding scheme. As a result of that number of Partial product reduce by one third factor. But the circuit complexity also increases as compare to previous version of booth multiplier.Radix-8 booth multiplier also lacks in most parameter like delay, speed from radix 4 booth multiplier due to the complexity of the circuit.Radix-8 booth multiplier which scan strings of four bits An algorithm similar to previous radix algorithm is made for radix8 booth multiplier [15],[16]. After encoding the multiplier the resulting partial product will be 0,y,+2y,+3y,+4y,- 4y,-3y,-2y,-y where y represent the multiplicand. We have to represent all the no. in 2's complement form [17],[18].

RESULTS AND DISCUSSION

SIMULATION RESULTS Unsigned X Unsigned Multiplication

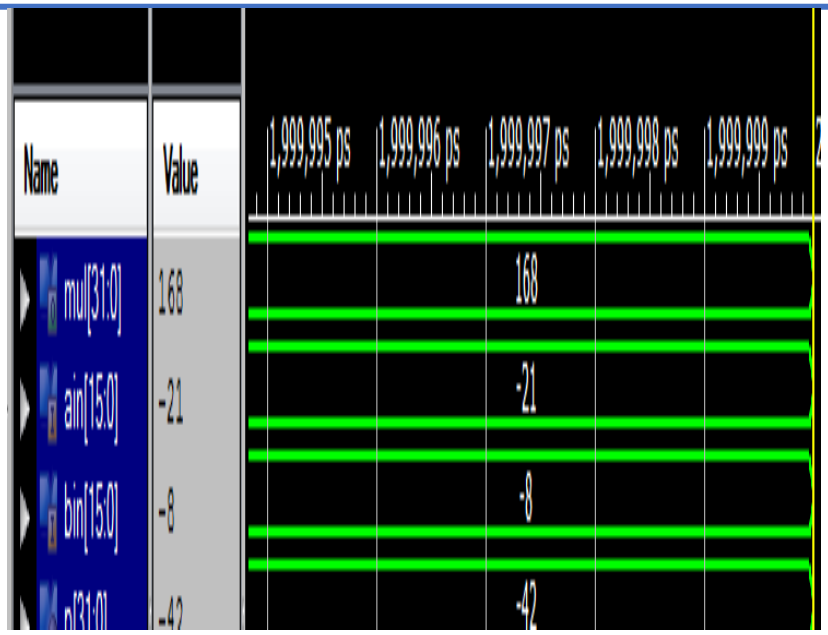


Ain=21 Bin=8 mul=168 Ain: Multiplicand Bin: Multiplier mul: Product

Multiplicand (MD) =0000000000010101

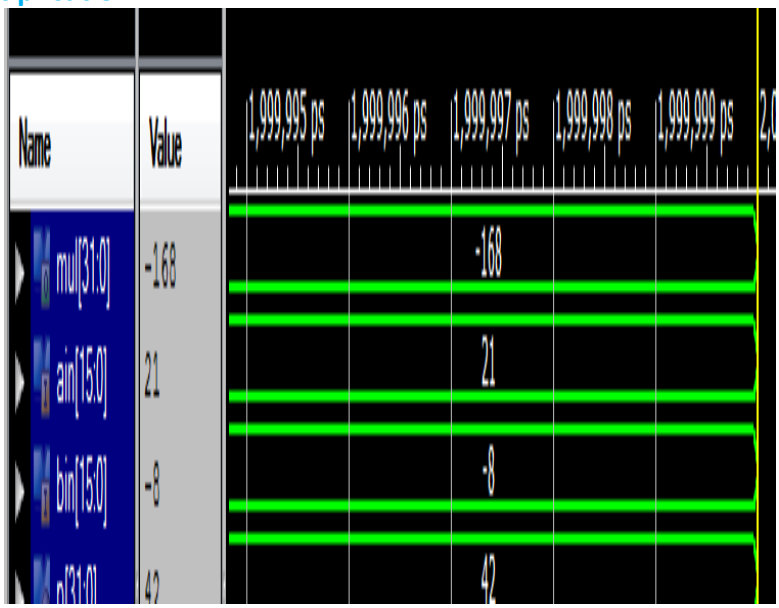
Multiplier (MR) =0000000000001000

Product (P) =0000000010101000 Signed X Signed Multiplication



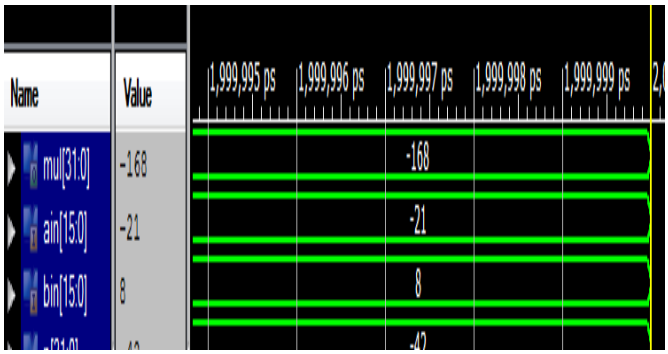
Ain = -21 Bin = -8 mul = 168 Ain: multiplicand Bin: multiplier mul: Product Multiplicand (MD) =1111111111101011
Multiplier (MR) =1111111111111000
Product (P) =0000000001010100

Unsigned X Signed Multiplication



Ain=21 Bin=-8 mul=-168 Ain: Multiplicand Bin: Multiplier mul: Product
Multiplicand (MD) =0000000000010101
Multiplier (MR) =1111111111111000
Product P) =1111111101011000

Signed X Unsigned Multiplication



Ain=-21 Bin=8 mul=-168 Ain: Multiplicand Bin: Multiplier mul: Product
Multiplicand (MD) =1111111111101011
Multiplier (MR) =0000000000001000
Product (P) =1111111101011000

SYNTHESIS RESULTS

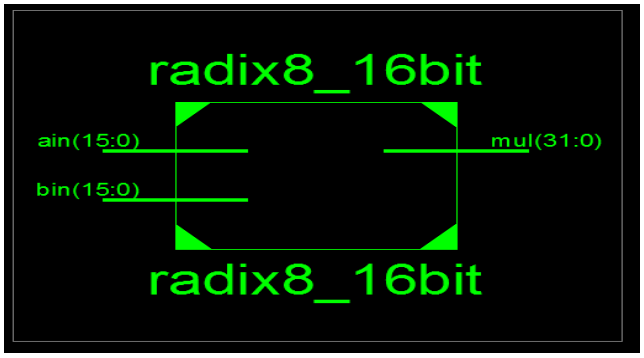


Fig 5. Block diagram of Multiplexer

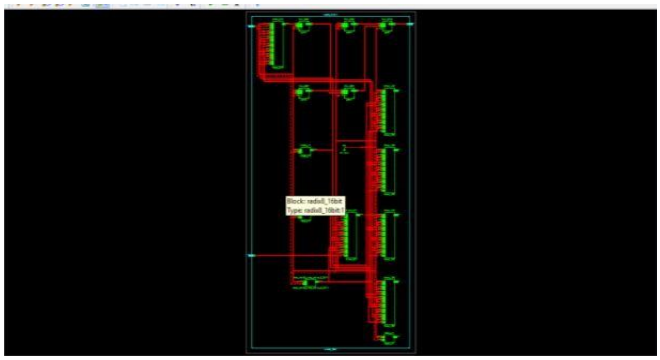


Fig 6. RTL schematic

COMPARISON TABLE:

S. No	Parameter	Radix 2	Radix 4	Radix 8
1	No. of Bits	16	16	16
2	No. of partial products	16	8	5
3	Delay	14.774 ns	13.602ns	12.550 ns
4	Frequency	67MHz	73MHz	80MHz
5	IOB	64	64	64
6	LUT	710	600	459

CONCLUSION

It has been performed the design and simulation of a 16x16 bit, radix-8 multiplier unit for signed and unsigned numbers using Xilinx 14.3 platform. 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 requires only 3 partial products one CSA and CLA to produce final products.

FUTURE SCOPE

The power saving may be increased if the following criterion is considered in the future low power VLSI design · Number of bits considered may be increased in the encoding scheme · Power can be reduced by improving the partial product compression ratio.

REFERENCES

1. C.Padma, P.Jagadamba, P. Ramana Reddy, Implementation of High Performance FFT Architecture for DSP Applications, International Journal of Advanced Science and Technology (IJAST) ISSN: 2005-4238, Vol.29, No.3(s), March 2020.
2. Honglan Jiang, Jie Han Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation IEEE Transactions on Computers, Volume. 65, 2016.
3. G Ganesh Kumar, Subhendu K Sahoo Implementation of A High Speed Multiplier for High-Performance and Low Power Applications IEEE Transactions, 2015.
4. Arish S R.K.Sharma Run-time reconfigurable multi- precision floating point multiplier design for high speed, low- power applications IEEE Tractions, 2015.

5. M. Vinod Kumar Naik, Mohammed Aneesh. Y, Design of Carry Select Adder for low power and High Speed VLSI Applications IEEE Transactions, 2015.
6. Jiun-Ping Wang, Shiann-RongKuang High-Accuracy Fixed- Width Modified Booth Multipliers for Lossy Applications IEEE Tractions, 2011.
7. Karthick, R and P, Meenalochini and Prabaharan, A.Manoj and Selvaprasanth, P. and M, Sheik Dawood, A Dumb-Bell Shaped Damper with Magnetic Absorber using Ferrofluids (December 2, 2019). International Journal of Recent Technology and Engineering (IJRTE), Volume-8, Issue-4S2, December 2019, Available at SSRN: https://ssrn.com/abstract=3517662 or http://dx.doi.org/10.2139/ssrn.3517662
8. Y.-H. Chen and T.-Y.Chang, A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers, IEEE Transactions on Circuits and Systems I, volume. 59, no. 3, pp. 594–603, 2012.
9. Meenalochini, P., and S. P. Umayal. "Comparison of Current Controllers on Photo Voltaic Inverters Operating as VAR Compensators." Journal of Electrical Engineering The Institution of Engineers, Bangladesh Vol. EE 38.
10. C.Padma and Y. Mahesh, Design of High Speed Modified Booth Encoded Parallel Multiplier, i-manager's Journal on Embedded Systems, Vol. 2 l No. 1 l February - April 2013.
11. CH. Pallavi and V. Swathi, An Efficient Carry Select Adder with Reduced Area Application, International Journal of Computer Engineering Science (IJCES) Volume 3 Issue 6 (June 2013) ISSN : 2250:3439.
12. Karthick, R and D, John Pragasam, Design of Low Power MPSoC Architecture using D-R Method (June 10, 2019). Asian Journal of Applied Science and Technology (AJAST), Volume 3, Issue 2, Pages 101-104, April -June 2019, Available at SSRN: https://ssrn.com/abstract=3401644
13. Karthick, R and Sundararajan, M., Optimization of MIMO Channels Using an Adaptive LPC Method (February 2, 2018). International Journal of Pure and Applied Mathematics, Volume 118 No. 10 2018, 131-135, Available at SSRN: https://ssrn.com/abstract=3392104
14. C. Liu, J. Han, and F. Lombardi, A low-power, high- performance approximate multiplier with configurable partial error recovery, in DATE, 2014, p. 95.
15. N. Zhu, W. L. Goh, and K. S. Yeo, An enhanced low- power high- speed adder for error tolerant application, in ISIC. IEEE, 2009.
16. Karthick, R and P, Meenalochini and R, Mohammed Abdullah, A Novel Hybrid Multilevel Inverter with Switch Reduction (June 6, 2020). International Journal of Advanced Science and Technology, Vol. 29, No. 4s, (2020), pp. 3018-3023, Available at SSRN: https://ssrn.com/abstract=3633572
17. Kuan-Hung Chen and Yuan-Sun Chu A Low-Power Multiplier with the Spurious Power Suppression Technique IEEE Transactions2007.
18. Kyung-Ju Cho, Kwang-Chul Lee, Jin-Gyun Chung, Design of Low-Error Fixed-Width Modified Booth Multiplier IEEE Transactions 2004.