

ANALYSIS OF MULTIPLIERS AND PROPOSAL OF A DESIGN OF VLSI ARCHITECTURE USING HYBRID MODIFIED BOOTH MULTIPLIER

S. Gajendran¹, G. Prabhakaran²

*P.G. scholar, Department of ECE, Nandha Engineering College, Erode, Tamilnadu, India¹
Assistant Professor, Department of ECE, Nandha Engineering College, Erode, Tamilnadu, India²*

Abstract: Compact Space and Minimal Power VLSI circuits are the standard for creating energy-efficient, high-performance, and small devices. Digital signal processing, microprocessors, and microcomputer applications all heavily rely on multipliers. In this design, we analysed and compared various multiplier types, including Wallace trees, arrays, and Booth multipliers. By using partial product reduction, multipliers can be designed to increase the creation of complex circuits and their analysis. Any algorithm for multiplying integers should aim to minimise the partial product summation. Among the most well-liked and effective algorithms is the Booth technique.

Keywords: Array, Wallace and Booth multipliers, Encoding, Decoding, CSA, MAC, CPA

1. Introduction

Multipliers are essential for processing digital signals in real time as well as for several other processes. The design of multipliers that offer increased speed, reduced power consumption, regular layout and, therefore, a small area, or even a combination of these in a single multiplier, has been attempted and is still being attempted by numerous researchers.

This makes the multipliers applicable for a variety of increased speed, reduced power, and compact VLSI prosecutions. The Booth technique and the Modified Booth algorithm, which minimise the partial products (1). It is evident that there is a desire for design patterns for digital systems that have low power consumption and high increment (2). We propose to redesign the existing. We suggest utilising a carry to redesign the becoming twofold multipliers. We propose to redesign the existing. By employing a carry look ahead adder, we suggest redesigning the being double multipliers. We obtain an efficient design in terms of detention optimisation by employing this method. The suggested approach produces a superior result in relative research where some of the variables are multipliers (3), (4). Verilog HDL is used to enforce the designs.

1.1 COMPARISON OF DIFFERENT MULTIPLIERS

An electronic hardware circuit used in computers or digital electronics is called a binary multiplier. Binary adders are used in its acquisition. The following are the guidelines for binary multiplication.

1. The product will be the same as the multiplicand and will just be copied down if the multiplier digit is 1.
 2. The product is zero if the multiplier digit is zero.
- Figure 1 illustrates the n X n multiplication algorithm

$$\begin{array}{r}
 Y = Y_{n-1} Y_{n-2} \dots Y_2 Y_1 Y_0 \text{ Multiplicand} \\
 X = X_{n-1} X_{n-2} \dots X_2 X_1 X_0 \text{ Multiplier} \\
 \hline
 \begin{array}{r}
 Y_{n-1} X_0 \quad Y_{n-2} X_0 \quad Y_{n-3} X_0 \dots Y_1 X_0 \\
 Y_{n-1} X_1 \quad Y_{n-2} X_1 \quad Y_{n-3} X_1 \dots Y_1 X_1 \quad Y_0 X_1 \\
 Y_{n-1} X_2 \quad Y_{n-2} X_2 \quad Y_{n-3} X_2 \dots Y_1 X_2 \quad Y_0 X_2 \\
 \dots \quad \dots \quad \dots \quad \dots \quad \dots \\
 Y_{n-1} X_{n-2} \quad Y_{n-2} X_{n-2} \quad Y_{n-3} X_{n-2} \dots Y_1 X_{n-2} \quad Y_0 X_{n-2} \\
 Y_{n-1} X_{n-1} \quad Y_{n-2} X_{n-1} \quad Y_{n-3} X_{n-1} \dots Y_1 X_{n-1} \quad Y_0 X_{n-1} \\
 \hline
 P_{2n-1} \quad P_{2n-2} \quad P_{2n-3} \dots P_2 \quad P_1 \quad P_0
 \end{array}
 \end{array}$$

1.1.A. Array multiplier

Array multipliers are widely recognised because of their regular structure. Add and shift is the algorithm that multipliers use. Multiplying the multiplicand by one multiplier bit yields each and every partial product. The partial products are added after being moved in accordance with their bit ordering. With a standard carry propagate adder, the addition may be completed.

A four-bit combinational multiplier consists of half adders, full adders, and AND gates. Two figures that are m and n bits wide, independently, allow us to generalise this. A series of m n AND gates operates in resembling to induce m n summands. For an n x n multiplier, thus, n half-adders, n (n - 2) complete adders, and 2n AND gates are demanded (5). n-1 adders, where n

is the multiplier length, are demanded. As demonstrated by this illustration, the addition is performed both in resembling and serially, despite the system's simplicity. One debit is that the tackle gets more complicated and takes up more space because so many gates are used, and it is less economical [6].

1.1.B. Wallace Tree multiplier

A digital circuit's attack performance that adds two double digits is called a Wallace tree multiplier. Every column that has a square measure of two bits uses half adders' square measure, while every column that has a square measure of three bits uses full adders' square measure. Any bit that is present in the column is moved to the next stage in the same column without any procedures. Repeat this technique until there are just two rows left. The square measures of the last two rows are value-added within the finish. Eight small pieces of affairs are typically obtained after completing all three stages. The circuit diagram for the Wallace Tree Multiplier using half and full adders were shown in Fig. 2. Straightforward, as demonstrated by this illustration

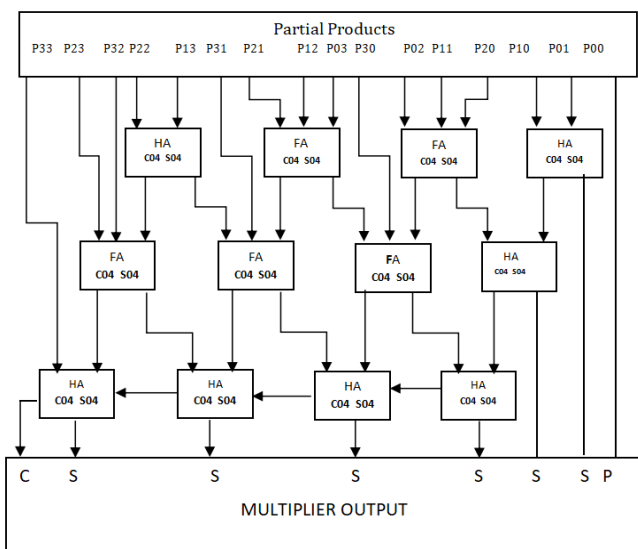


Fig. 2 Internal blocks of Wallace Tree Multiplier

In the first step, partial products are created by multiplying each bit of the multiplicand by each bit of the multiplier. The length of the multiplicand is therefore used to compute the number of partial products. [7], for each column with two bits, a half adder is utilised, and for each column with three bits, a full adder. Any single bit in the column is transferred, without any processing, to the next step in the same column. Every step of the reduction process is repeated until there are just two left. In [8] the final step involved adding the two remaining rows. We obtain an output of 8 bits once all three processes have been completed. The example scheme is displayed in Figure 2.

This figure illustrates the arrangement of the Wallace tree, which has carried save adders. A carry save adder (CSA) generates a second result while the adder is still complete by bringing out the carry output of each bit. Enhancement of worst-case path delay is the aim of CSA [9]. It uses a 4-bit to implement CSA. The second addition uses the carry input from the first addition. Increasing the maximum frequency is one of CSA's benefits [10].

A sequence of complete adders, each one a single bit, is used in the traditional CSA tree to gradually reduce partial product bits with numerous inputs at the same bit position to a final sum and carry pair. After the output, we will have a sum and carry, which a carry-propagate adder (CPA) must add. Radix-8 recoding is faster than radix-4 recoding, which utilises $n/2$, because it reduces the partial products to $n/3$ for n bits [11].

1.1.C. Booth multipliers

A crucial component of digital signal processing (DSP) systems is booth multipliers. Recursive and transversal filter implementations make use of them. In contrast to add shift algorithms, which require less hardware and perform worse, conventional array multipliers, reach relatively acceptable performance but require a vast silicon area. The operation is reduced to a partial product summation in any multiplication algorithm.

Each partial product indicates a multiplicand multiple that ought to be included in the outcome [12]. In order to achieve a speed advantage over conventional multiplier architectures, the Booth multiplier uses the Booth encoding technique [13] to limit the partial products by taking into account two multiplier bits at a time. A clever method for multiplying signed numbers is Booth's Algorithm. The ability to add and subtract is the first step. [14]

1.2. BOOTH ENCODING MULTIPLIER:

While studying crystallography in 1950, Andrew Donald Booth created the Booth algorithm. The Booth multiplier achieves a speed advantage over existing multiplier architectures by using the Booth encoding technique to limit the partial products. An intelligent method for multiplying signed numbers is Booth's algorithm. Being able to add and subtract is the first step. It uses the encoding of 2's complement to handle signed binary multiplication [15]. The process by which operand signs are stored in auxiliary circuits becomes more intricate as a result. It is a highly used algorithm for multiplication of signed numbers that uniformly takes into account both positive and negative numbers [16].

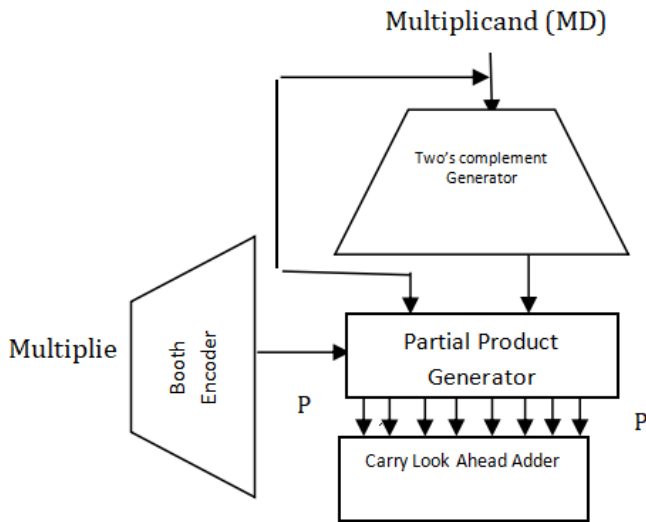


Figure.3 Architecture of Booth Multiplier

1.2.A. Radix 4 Booth Multiplier

The Booth technique [17] is an effective algorithm for multiplication of signed numbers. It can handle more than one bit of the multiplier in each cycle by employing high radix multiplication [18]. The Radix-2 algorithm's drawbacks are all addressed by the Radix-4 modified Booth algorithm.

Modified Booth Multiplier

The enhanced Booth multiplier is known as the modified Booth multiplier [19]. In the Modified Booth, the quantity of partial items produced is cut in half. To achieve the same results, multiply every second column by ± 1 , ± 2 , or 0 rather than shifting and adding each column of the multiplier term and then multiplying by 1 or 0.

1 1 1 0 0 0 1 1 0

Bit

pairing as per bit encoding

The multiplicand is encoded using the multiplier bits by the radix-4 booth encoder. It will use the overlapping approach to compare three bits at a time. The first block, as shown in Fig. 3, assumes a zero for the third bit and uses only two of the multiplier's bits. Truncating starts at the LSB.

Table 1 displays Radix-4 booth encoder's operational state. There are eight various kinds of states in it, and we can get the results of multiplying the multiplicand by 0, -1, and -2 in that order.

Table 1. Booth recording table for Radix - 4

Multiplier Bits block			Recorded 1 bit pair		2 Bit booth	
i+1	I	i-1	I+1	I	Multiplier value	Partial product
0	0	0	0	0	0	0X
0	0	1	0	1	1	1X
0	1	0	1	-1	1	1X
0	1	1	1	0	2	2X
1	0	0	-1	0	-2	-2X
1	0	1	-1	1	-1	-1X
1	1	0	0	-1	-1	-1X
1	1	1	0	0	0	0X

Radix-4 Booth algorithm for Booth Encoding is given below:

- 1) If necessary, change the sign bit one position to ensure that n is correct.
- 2) Place a zero to the multiplier's right of the LSB.
- 3) In line with each vector's value. There will be 0, X, -X, 2X, or -2X for each partial product.

1.2.B. Radix 8 Booth Encoding :

The main difference between radix 8 Booth recoding and radix 4 Booth recoding is the use of quartets of bits rather than triplets. Radix 8 reduces the partial products to $n/3$. Our multiplier, however, is made to be modified at the earlier phases of the adder. With the exception of employing quartets of bits rather than triplets, the algorithm for Radix-8 Booth recoding is the same as for Radix-4 Booth recoding.

Table 2. Booth Algorithm Radix 8 truth table

Multiplier Bits				Operation on Multiplicand
A	B	C	D	X
0	0	0	0	0X
0	0	0	1	+1X
0	0	1	0	+1X
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

The Booth multiplier consists of three basic unit cells:

Encoder

The encoder circuit's general structure is the double law that the affair lines produced in response to the input value.

Decoder

An array of inputs and labors in a sense circuit known as a decoder converts encoded inputs into encoded labors indeed when the input and affair canons are different. With n possible labors and 2n inputs, it presents the decoder data.

12-bit adder

The 4-bit carry look ahead adder with the 4-bit full adder comprises the 12-bit adder circuit.

A complete adder is a three-bit column addition circuit. Given that the final addition time greatly increases the multiplier's critical path, it is obvious that the quickest method ought to be applied [20]. The current structure is changed by replacing the final complete adder stage with a carry look ahead adder.

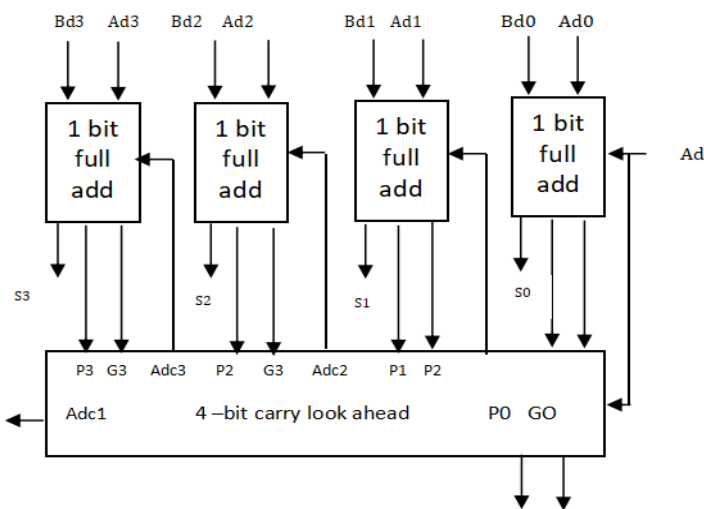


Fig.4. 4-bit Carry look ahead adder

4-bit Carry look ahead Adder

Generating and propagating carries are ideas used in carry look-ahead logic. In order to ascertain the carry in advance, the circuit may "pre-process" the two integers being added. By doing this, there's no longer a need to stay for the addition to actually take effect. Every function that generates and propagates is stated in terms of the entire adders' inputs, rather than waiting for the carry to ripple through each stage; all of the labors are now incontinently available (21). After making a few

minor adjustments, the basic 4-bit generalised Carry Look Ahead circuit may be connected to the 4-bit Ripple Carry Adder that we initially utilised.

Both of radix-4 and radix-8 multipliers were designed by using the Modified Booth Algorithm. The Radix-8 Modified Booth Multiplier outperforms the Radix-4 Modified Booth Multiplier. Thus, the radix-8 multiplier had a shorter time period than the radix-4 multiplier.

2. IMPLEMENTATION OF MAC UNIT FOR DSP PROCESSING

The system's energy consumption and presto operation are measured by the MAC unit. The MAC unit is a pivotal element of the signal processing unit. The final step in that design is to build a multiplier accumulator (MAC) unit for high-speed digital signal processing.

The sum of an accumulator, multiplier, and adder is known as MAC. The multiplier factor block performs the addition operation after we recoup the multiplier and accumulator input from the memory address. Data is subsequently transferred to the adder, which compiles all of the data before storing it in a memory position. Several addition styles, analogous to bit-quotidian, journal-parallel or full-similar approaches, can be used to produce partial products. For this, the Booth or modified Booth algorithms are generally employed.

The architecture of the multiplier is separated into three stages: partial product reduction, partial product generation, and the supplementary reduced partial product stage. Multiplications have historically been done using the shift and add algorithm (22).

2.1 SUGGESTED MAC UNIT

A modified Booth-encoded hybrid multiplier and a parallel hybrid adder will be used in the suggested design. The design goal of the proposed MAC (Multiply and Accumulate) unit is to achieve efficiency in both area and power. Both a hybrid multiplier and a hybrid adder are employed, along with a changed encoding scheme.

Modified booth multiplier

The 16-bit multiplier (B(15:0)) and multiplicand (A(15:0)) are the system's inputs. The Modified Booth garbling block processes these inputs in order to reduce the partial products. Booth garbling does this by combining and garbling the multiplier's bits in a way that decreases the total number of operations required.

The encoding process is followed by the pipelining block for partial products receiving the

encoded outputs. This simultaneous processing greatly increases the multiplication process's total speed. Another important part of this architecture is conditional gating. Power consumption is efficiently managed by this block, which regulates data flow according to predetermined criteria. The output is the 32-bit product result of the multiplication.—is the last element and is denoted by the symbol $Y[31:0]$. Following completion of each pipeline stage, the end result is the cumulative sum of the PPs.

3. SIMULATION AND SYNTHESIS RESULTS

The developed MAC design is comprised of three phases: design entry, simulation, and synthesis. The suggested design has been built in Verilog, and its functioning has been confirmed using the Cadence NCSim simulator. After the functional verification is finished, the optimised netlist is inspected using the RTL model in the Genus Synthesis Tool to obtain data on time, power, area, and other variables. Referred papers are compared with these areas, power, etc.

4. CONCLUSION

The suggested MAC (Multiply-Accumulate) unit is very efficient in terms of both power and area. The design essentially combines a Booth multiplier with a mongrel adder, with each optimised for its own capabilities. The mongrel multiplier uses a modified garbling strategy to reduce area while increasing speed, whilst the mongrel adder, built using a mix of CSA and CLA, plays an important role in minimising power usage and further lowering area. The suggested MAC unit works better than conventional designs, as supported by previous articles, as demonstrated by the whole armature, which was encoded in Verilog and validated using the meter NCSim simulator. Finally, as confirmed by both simulated and synthesised findings, the armature of this suggested MAC unit exhibits an optimising balance of power & area efficiency.

5. REFERENCES

- [1] Prasanna Raj P, Rao, Ravi, —VLSI Design and Analysis of Multipliers For Low Power, Intelligent Information Hiding and Multimedia Signal Processing, Fifth International Conference, pp.: 1354-1357, Sept. 2009.
- [2] Anantha P. Chandrakasan, Samuel Sheng, Robert W. Brodersen" Low-Power CMOS digital design," IEEE Journal of Solid State Circuits, Vol. 27, No. 4, 1992.
- [3] Kelly Liew Suet Swee, Lo Hai Hiung, "Performance Comparison Review of 32-Bit Multiplier Designs," IEEE, 2011.
- [4] Hasan Krad, Yousif Al-Taie Aws, "Performance Analysis of a 32-Bit Multiplier with a Carry-Look-Ahead Adder and a 32-bit Multiplier with a Ripple Adder using VHDL," Journal of Computer Science 4 (4): pp. 305-308, 2008.
- [5] Shaik Nasar, K. Subbarao, "Design & Implementation of MAC Unit Using Reversible Logic," International Journal of Engineering Research and Applications Vol. 2, Issue5, pp. 1848-1855, 2012.
- [6] Sumit Vaidya, Dandekar. D, "Delay-power performance comparison of multipliers in VLSI circuit design," International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, 2010.
- [7] B. Lamba and A. Sharma, "A review paper on different multipliers based on their different performance parameter", 2nd International Conference on Inventive Systems and Control, pp 324-327, 2018.
- [8] Bhawna Singroul, Pallavee Jaiswal, "A review on performance Evaluation different digital multiplier in VLSI using VHDL", International journal of Engineering research & technology, vol-7, issue-5 , 2018.
- [9] Sumit Vaidya, Deepak Dandekar, "A review on delay performance comparison of multiplier in VLSI circuit design", International journal of computer network & communication, Vol 2, issue 4, pp 47-55, 2010
- [10] Soniya, Suresh kumar, "A review of different types multiplier and multiplier accumulator unit", International journal Emerging Trends and technology in computer science, vol-2, issue-4, pp 364-368, 2013
- [11] lakshmanan, m. othman, m.a.m. ali, "design and characterization of parallel prefix adders using fpgas," journal of computers, vol. 5, no. 10, october 2012.
- [12] J.A. Hidalgo, V. Moreno-Vergara, O. Oballe, A. Daza, M.J. Martín-Vázquez, A.Gago, "A Radix-8 multiplier design for specific purpose"@2011.
- [13] Kelly Liew Suet Swee, "Performance And Comparison Review Of Radix Based Multiplier Design", International Conference Of Intelligent And Advanced System, 2012.
- [14] Depth:In More Booth_s Algorithm, staff.ustc.edu.cn/~han/CS152CD/Content/COD3e/in moredepth/IMD3 -Booths-Algorithm.pdf -

[15] Abenet Getahun, –Booth Multiplication Algorithm,|| Fall 2003 CSCI 401

[16] A. D. BOOTH, “A signed Binary multiplication technique”, in journal of Mech. APPL. Math, Oxford University press, vol-4, pp 236-240,1951.

[17] Rajput.S, Shukla.R, Praveen.P, Anand.A, “Implementation of high speed and low power hybrid adder based novel radix-4 booth multiplier”, Communication System Network Technologies, International Conference,2013.

[18] Premananthan.G, 2Amudha.K, 3 Sreenath.G, “A Faster Carry Save Adder in Radix-8 Booth Encoder Multiplier” ,International Journal of VLSI & Signal Processing Applications, Vol. 2,Issue2, April 2012, ISSN 2231-3133, (171-175).

[19] Kim, S. and Cho, K. Design of high-speed modified booth multipliers operating at GHz ranges. World academy of science, Engineering and Technology 61 (2010), pp. 1–4.

[20] Vojin G. O, D. Villeger, Simon S. L, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," IEEE Transactions on computers, Vol. 45, No. 3, 1996.

[21] A. Anand Kumar, "Fundamentals of Digital Circuits," pp. 242-245 PHI Learning Private Ltd. New Delhi, 2008.

[22] Bhavya Lahari Gundapaneni, JRK kumar Dabbakutti, “ A review on Booth Algorithm for the design of multiplier”, International journal of innovative technology and Exploring Engineering , vol-8, issue-7, pp 1506-1509 , 2019