**ABSTRACT:**

This report describes implementation of array multiplier and booth radix 4 multiplier and comparison is made between them. Modified Booth’s algorithm employs both addition and subtraction and also treats positive and negative operands uniformly. No special actions are required for negative numbers. Modified Booth multiplication algorithm is designed using high speed adder. High speed adder is used to speed up the operation of Multiplication. Designing of this algorithm is done by using VERILOG and simulated using Xilinx ISE 14.7 software.

The algorithm of booth multiplier furnishes a level to formulate a multiplier with greater efficacy & speed as compared to array multiplier. This algorithm gives a better level of encoding in the commencing stage of multiplication of 8 & 4-radix. Also outcomes of delay & LUTs are improvised by deploying pipelining.

Algorithms for the array multiplication of two n- bit binary numbers by an iterative array of logic cells are also discussed. The regular interconnection structures of the multiplier array cell elements, which are ideal for VLSI implementation, are described. In this paper, a modified full adder and half adders are proposed to achieve low power consumption of multiplier. To analyze the efficiency of proposed design, the conventional array multiplier structure is used. The designs are developed using Verilog HDL and the functionalities are verified through simulation using Xilinx

**INTRODUCTION TO THE TOPIC:**

Multipliers are widely used in arithmetic units of microprocessors, multimedia and digital signal processors; moreover, high performance and low power multipliers are in high demand for embedded systems. It is becoming extremely difficult to further improve performance and reduce the power consumption of multipliers under the requirement of full accuracy.

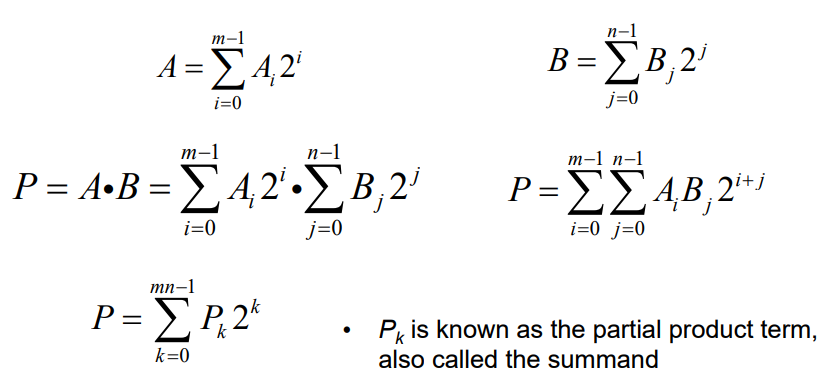
**ARRAY MULTIPLIER:**

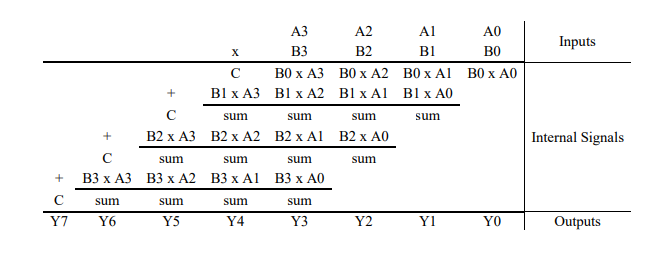
 Array multipliers are the most popular kinds of multiplier. Multiplication consists of three steps: 1) generation of partial products, 2) accumulation of all partial production to get only two rows at the end and last two rows of partial products are added by using a carry propagation adder. 3) Last step is to efficiently add the two rows of partial products which can be fastened by using appropriate adder structure (full adder or half adder).

ALGORITHM:

Array multiplier is well known due to its regular structure. Multiplier circuit is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added. The addition can be performed with normal carry propagate adder. N-1 adders are required where N is the multiplier length. [1]

There are mn summands that are produced in parallel by a set of mn AND gates – n x n multiplier requires n(n-2) full adders, n half-adders and n2 AND gates – Worst case delay would be (2n+1)td.

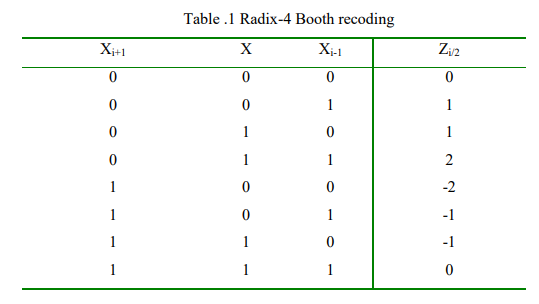




**BOOTH RADIX 4 MULTIPLIER:**

Booth's algorithm examines adjacent pairs of bits of the 'N'-bit multiplier *Y* in signed two's complement representation, including an implicit bit below the least significant bit, *y*−1 = 0. For each bit *yi*, for *I* running from 0 to *N* − 1, the bits *yi* and *yi*−1 are considered. Where these two bits are equal, the product accumulator *P* is left unchanged. Where *yi* = 0 and *yi*−1 = 1, the multiplicand times 2*i* is added to *P*; and where *y*i = 1 and *y*i−1 = 0, the multiplicand times 2*i* is subtracted from *P*. The final value of *P* is the signed product.

The representations of the multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at *i* = 0; the multiplication by 2*i* is then typically replaced by incremental shifting of the *P* accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest *N* bits of *P*.[2]



ALGORITHM:

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward [arithmetic shift](https://en.wikipedia.org/wiki/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**.

1. 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).
   1. A: Fill the most significant (leftmost) bits with the value of **m**. Fill the remaining (*y* + 1) bits with zeros.
   2. S: Fill the most significant bits with the value of (−**m**) in two's complement notation. Fill the remaining (*y* + 1) bits with zeros.
   3. P: Fill the most significant *x* bits with zeros. To the right of this, append the value of **r**. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of *P*.
   1. If they are 01, find the value of *P* + *A*. Ignore any overflow.
   2. If they are 10, find the value of *P* + *S*. Ignore any overflow.
   3. If they are 00, do nothing. Use *P* directly in the next step.
   4. jnIf they are 11, do nothing. Use *P* directly in the next step.
3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let *P* now equal this new value.
4. Repeat steps 2 and 3 until they have been done *y* times.
5. Drop the least significant (rightmost) bit from *P*. This is the product of **m** and **r**.[3]

**LITERATURE SURVEY RELATED TO THE TOPIC:**

**ARRAY MULTIPLIER**

With an array multiplier two binary numbers will be multiplied by using of an array of half adders and full adders. Simultaneously addition of the different product terms is done in this array. By using an array of AND gates, the partial product terms are formed. Following this an array of AND gates, the adder array is used. The hardware structure for an pxq bit multiplier is described as (pxq) AND gates (p-1) q adders .Here q half adders and (p-2).q Full adders.

Array multiplier does the multiplication process in traditional way. It looks like regular structure. Hence wiring and the layout are done in a much simplified manner. Add and shift algorithm is employed in an array multiplier. Implementation of this multiplier is simple but it requires larger area, with considerable delay also [4]. Instead of Ripple Carry Adder (RCA), in this multiplier Carry Save Adder (CSA) is used for adding each group of partial product terms, because RCA is the slowest adder among all other types of adders available. In case of multiplier with CSA, partial product addition is carried out in Carry save form and RCA is used only in final addition.

In this algorithm, no waiting is necessary until all the partial products have been formed before summing them. As soon as the partial products formed immediately the addition of partial product can be done [5]. The major advantage of the array multiplier is that it has a regular structure. Another advantage of the array multiplier is its ease of design for a pipelined architecture. Major limitation of array multiplier is its size. As operand sizes increase, arrays grow in size at a rate equal to the square of the operand size. It comes under conventional multiplier.

**BOOTH RADIX 4**

In this work [6] different multiplier architectures (array multiplier and booth radix 4 multiplier) are implemented in Xilinx FPGA and compared for their performance It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. To overcome the limitation of array multiplier the speed of the multiplier is increased by the booth algorithm.

Booth algorithm reduces the number of partial products. Here, the multiplier considers two number of bits at a time for the multiplication process. The multiplication process for both signed and unsigned numbers can be done in this booth multiplier. This multiplier considers the 2's compliment of the given multiplicand and multiplier. It is based on radix-2 computation. In add-shift operation each multiplier bit multiply with the multiplicand and to be added to the partial product. For very large multiplier, a large number of multiplicands can be added. In this multiplier, number of additions can decide the multiplier delay.

Booth algorithm can easily reduce the no. of multiplicand multiplies. For a n-bit number can be represented as n/2- digit radix 4 number, a n/3- digit radix 8 number and so on. Major limitation of array multiplier is its size. As operand sizes increase, arrays grow in size at a rate equal to the square of the operand size, hence speed of multiplier reduces .In order to increase the speed of multiplier booth algorithm is used. The Booth multiplier makes use of Booth encoding algorithm in order to reduce the number of partial products by considering two bits of the multiplier at a time, thereby achieving a speed advantage over other multiplier architectures. This algorithm is valid for both signed and unsigned numbers. It accepts the number in 2's complement form, based on radix-2 computation [6].

The low power consumption quality of booth multiplier makes it a preferred choice in designing different circuits. By implementing both Radix-2 & Radix -4 multiplier using booth algorithm their computation speed increases so much.

**REFERENCES:**

[1].<http://eee.guc.edu.eg/Courses/Electronics/ELCT706%20Microelectronics%20Lab/winter%202014/Multipliers.pdf>

[2]. Chi-hau Chen (1992). [*Signal processing handbook*](https://books.google.com/books?id=10Pi0MRbaOYC&pg=PA234). CRC Press. p. 234. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*978-0-8247-7956-6*](https://en.wikipedia.org/wiki/Special:BookSources/978-0-8247-7956-6).

**[3].** [**https://en.wikipedia.org/wiki/Booth%27s\_multiplication\_algorithm**](https://en.wikipedia.org/wiki/Booth%27s_multiplication_algorithm)

[4] S.P.Pohokar, R.S.Sisal, K.M.Gaikwad, M.M.Patil, Rushikesh Borse “Design and Implementation of 16x16 Multiplier using vedic mathematics” International Conference on Industrial Instrumentation and control(ICIC) College of Engineering Pune, India, May 28-30,2015

[5] V.Priyanka Brahmaiah ,L.Dhrma Teja, Dr Y.Padma sai “Study on comparison of various multipliers” International Journal of Electronics and communication engineering & technology(IJECET), ISSN 0976-6464 volume 4 Issue 5 September – October, 2013 pp-132-142.

[6] Ruchi Sharma “Analysis of Different Multiplier with Digital Filters Using VHDL Language” International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-1, October 2012.