International Journal of Recent Trends in Engineering & Research (IJRTER)
Conference on Electronics, Information and Communication Systems (CELICS'18)
Special Issue; March - 2018 [ISSN: 2455-1457]
DOI : 10.23883/IJRTER.CONF.02180328.031.MFPRK

Performance Comparison of Radix-2 and Radix-4 by Booth Multiplier

Sakthivel B[1],M.Nivedhaa[2],A.Manimegalai[3],M.Vignesh[4],N.Satheesh[5]
Assistant professor[1],Students[2,3,4,5]
Department of Electronics and Communication Engineering
COIMBATORE INSTITUTE OF ENGINEERING AND TECHNOLOGY
COIMABATORE-641109

Abstract— This paper describes implementation of Radix-4 modified Booth multiplier and this implementation is compared with Radix-2 booth multiplier. Modified Booth's algorithm employs both addition and subtraction and also treats positive and negative number. No special actions are required for negative numbers. High speed adder is used to speed up the operation of multiplication. It presents the implementation and comparison of high speed multiplier which depends on booth encoding. In this, we compare the performance of Radix-2 and Radix-4 based on booth multiplier. Designing of this algorithm is done by using VHDL and simulated using Xilinx ISE 9.li software has been used and implemented on FPGA xc3s50-5pq208.

Keywords—booth multiplier, modified booth multiplier, CLA, VHDL

I. INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters, Microprocessor, digital signal processors, etc. A system performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. It is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue however; area and speed are usually conflicting constraints so that improving speed results mostly in larger areas.

Signed multiplication is a careful process with unsigned multiplication there is no need to take the sign of the number into consideration. In signed multiplication the same process cannot be applied because the signed number is in a 2"s complement which would yield an incorrect result if multiplied in a similar fashion to unsigned multiplication. Booth algorithm is also used for high speed multiplication.

Booth Multiplier reduces number of iteration step to perform multiplication as compare conventional steps. Booth algorithm „scans" the multiplier operand and skips chains of this algorithm can reduce the number of additions required to produce the result compared to conventional multiplication algorithm, where each bit of the multiplier is multiplicand and the partial products are aligned and added together. More interestingly the number of additions is data dependent, which makes this algorithm.

COMMON FEATURES OF MULTIPLIERS

(*i*) **Counterflow Organization:** A novel multiplier organization is introduced, in which the data bits flow in one direction, and the Booth commands are piggybacked on the acknowledgments flowing in the opposite direction.

DOI : 10.23883/IJRTER.CONF.02180328.031.MFPRK

(*ii*) **Merged Arithmetic/Shifter Unit:** An architectural optimization is introduced that merges the arithmetic operations and the shift operation into the same function unit, thereby obtaining significant improvement in area, energy and speed.

(*iii*) **Overlapped Execution:** The entire design is pipelined at the bit-level, which allows overlapped execution of Proceedings of multiple iterations of the Booth algorithm, including across successive multiplications. As a result, both the cycle time per booth iteration, as well as the overall cycle time per multiplication are significantly improved.

(*iv*) **Modular Design:** The design is quite modular, which allows the implementation to be scaled to arbitrary operand widths without the need for gate resizing, and without incurring any overhead on iteration time.

(*v*) **Precision-Energy Trade-Off:** Finally, the architecture can be easily modified to allow dynamic specification of operand widths, i.e., successive operations of a given multiplier implementation could operate upon different word length

RADIX-2:

Our multiplier is of the iterative Radix-2 booth multiplier type, implemented using asynchronous circuits. An iterative implementation was chosen, as opposed to a combinational array type, for higher area efficiency. A booth implementation was chosen so as to uniformly handle signed as well as unsigned operands. Booth algorithm gives a procedure for multiplying binary integers in signed -2‟s complement representation.

Working from LSB to MSB replace each 0 digit of the original number with a 0 in the recorded number until a 1 is encountered. When 1 is encountered insert a 1 at that position in the recorded number and skip over any succeeding 1‟s until a 0 is encountered. Replace that 0 with a 1 and continue.

A. BOOTH RECODING TABLE(Radix-2)

Q_n	Q_{n-1}	RECODED BITS	OPERATION PERFORMED
0	0	0	Shift
0	1	+1	Add
1	0	-1	Sub
1	1	0	Shift

BOOTH ALGORITHM:

1. add 0 to right of LSB of multiplier and look at rightmost of multiplier to make pairing of 2 bits from right to left and mark corresponding multiplier value.

2.00 or 11: do nothing.

3.01: marks the end of a string of 1‟s and add multiplicand to partial product (running sum).

4.10: marks the beginning of a string of 1‟s subtract multiplicand from partial product.

EXAMPLE:

M 0110 +6
Y 0010 +2

Accumulator	Y	Y_{n-1}	Z	Operations
0000	0010	0	0	
0000	0001	0	-1	Shift
1010 1101	0001 0000	0 1	1	A->A-M Shift
0011 0001	0000 1000	1 0	0	A->A+M shift
0000	1100	0		

DOI : 10.23883/IJRTER.CONF.02180328.031.MFPRK

DRAWBACKS OF RADIX-2:

The number of add and subtract operands and the number of shift operation become variable and become inconvenient in designing parallel multiplier.

To overcome this problem we are using Radix-4 booth multiplier.

The algorithm becomes inefficient when there are isolated 1s.

RADIX-4:

Speeding up multiplication using booth algorithm can be achieved by recording the multiplier in a higher radix than radix-2. Higher radix recoding means greater number of multiplier bits are inspected and eliminated per cycle resulting in less number of cycles required to obtain the product.

The negative values are made by taking the 2"s complement and carry-look-ahead (CLA) fast adders are used. The multiplication is done by shifting one bit to the left. Thus, in any case, in designing n-bit parallel multipliers, only n/2 partial products are generated. The number of summands are very low in this method.

BOOTH RECODING TABLE(Radix-4):

SELECT LINE (Encoding)	PARTIAL PRODUCTS (Operation)
000	0
001	1*Multiplicand
010	1*Multiplicand
011	2*Multiplicand
100	-2*Multiplicand
101	1*Multiplicand
110	-1*Multiplicand
111	0

BOOTH ALGORITHM:

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,-y,+2y or -2y.

EXAMPLE:

00 1 11 0 (21)10 Multiplicand
0 1 0 1 0 (14)10 Multiplier

+1 -1+1 -1+1 -1 Booth recoded multiplier
+1 +1 +1 Bit pair recoded multiplier

0 0 0 0 0 0 0 0 1 1 1 0 (14*1)10
0 0 0 0 0 0 1 1 1 0 0 0 (14*4)10
0 0 0 0 1 1 1 0 0 0 0 0 (14*!6)10

0 0 0 1 0 0 1 0 0 1 1 0 (294)10 Product

COMPARISON OF RADIX-2 AND RADIX-4:

	RADIX-2	**RADIX-4**
Minimum period:	5.454ns	4.750ns
Minimum input arrival time before clock:	7.936ns	4.014ns
Maximum output required time after clock:	6.216ns	6.205ns
Number of inputs and outputs:	16	16

Special Issue; March - 2018 [ISSN: 2455-1457]
DOI : 10.23883/IJRTER.CONF.02180328.031.MFPRK

CONCLUSION:

Radix-2,4 Booth multipliers are implemented here, the complete process of the implementation is giving higher speed of operation. compared to Radix-2 the radix-4 less area and time consuming. speed is high in radix-4.

REFERENCES:

[1] H.Lee," A High-Speed Booth Multiplier",ICCS,(2002).

[2] Vincent P.Heuring, Harry F.Jordon (2003),Computer Systems Design and Architecture, Pearson Education, Singapore.

[3] D. Kudeeth., "Implementation of low-power multipliers",Journal of low-power electronics, vol. 2, 5-11, (2006).

[4] Y.N. Ching, "Low-power high-speed multipliers", IEEE Transactions on Computers,vol. 54, no. 3,pp.355-361,2005.

