IJIEMR Transactions, online available on 15th FEB 2019. Link :
http://www.ijiemr.org/main/index.php?vol=Volume-08&issue=ISSUE-02

Title: **AN EFFECTIVE IMPLEMENTATION OF HIGH SPEED RADIX-4 AND RADIX-8 MULTIPLIER BY VERILOG HDL**

Volume 08, Issue 02, Pages: 39–44.

Paper Authors

SAISANDHYARANI SINGANAMALA

AP IIIT, Rajiv Gandhi University of Knowledge Technologies, INDIA

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

AN EFFECTIVE IMPLEMENTATION OF HIGH SPEED RADIX-4 AND RADIX-8 MULTIPLIER BY VERILOG HDL

SAISANDHYARANI SINGANAMALA
AP IIIT, Rajiv Gandhi University of Knowledge Technologies, INDIA

Abstract: A quick and energy productive multiplier is constantly required in gadgets industry particularly DSP, picture handling and number juggling units in microchips. Multiplier is, for example, imperative component which contributes significantly to the complete power utilization of the framework. On VLSI level, the zone likewise turns out to be very vital as more zone implies more framework cost. Speed is another key parameter while planning a multiplier for a particular application. Presently a-days the power utilization is the serious issue for the electronic gadgets. In this way, to structure the incorporated circuit, to play out the low power, less occupation zone and rapid at the same time. This paper present to structure the superior parallel radix-4 and radix-8 multiplier by utilizing altered booth algorithm. The structure for configuration is mxn duplication. Where, m and n reach up to 8bits. Convey the speed of administrator. This plan procedure is done in verilog HDL and reenactment by utilizing model sim simulator.

Keywords: VLSI, Verilog HDL, Area, Power Consumption.

1. INTRODUCTION

Multiplication is a standout amongst the most widely recognized tasks in computerized flag preparing. Duplicating two numbers is a computationally serious assignment on programming, which requires fundamental calculation time just as equipment assets [1]. In correspondence frameworks, multiplication of complex numbers is regularly required, which makes the task much all the more requesting as far as computational time and equipment assets. In this work, we think about various multiplication algorithms for 16-bit complex multiplication application. Multiplication of complex numbers can be decreased into a few genuine number activities. Two structures, one utilizing four multipliers and the other utilizing three multipliers, have been utilized broadly previously. The four-multiplier design is quicker, though the three-multiplier engineering requires less assets. Asset utilization can be additionally advanced with lesser-known two-multiplier engineering exhibited by Hemnani et al. [2] This work focuses on creating complex number multipliers, which comprise of genuine number multiplication and expansion activities. Genuine number multiplication on equipment is basically a progression of increases. First equipment multipliers utilized move and include algorithm framing and amassing halfway items one by one. The activity should be possible quicker with an exhibit multiplier structure, which shapes incomplete items in parallel. Besides, all halfway items are aggregated at the same time. [1] Multiplication can be part in three phases. To start with, all potential fractional results of a multiplicand are gotten and required ones are chosen dependent on the bits of the multiplier. At that point, chose fractional items are moved and summed up. This is typically performed in two phases. The chose halfway items are first packed into two numbers, which are then included with a quick last viper [1]. In this work, we keep the pressure and last expansion comparative for every execution, and plan the incomplete item age separatedly for every design. Booth proposed an equipment multiplication algorithm as of now in 1951 [3]. It took ten additional years until MacSorley distributed an equipment multiplier [4]. A couple of years after the fact, Wallace and Dadda enhanced the speed of the multiplication with their proposition [5, 6]. The reason for this work is to execute and analyze four distinct multipliers: a radix-4 Booth encoded, radix-8 nonrecoded, radix-8 Booth encoded, and radix16 Booth encoded models. Albeit all the previously mentioned structures are focused for ASIC (application-explicit coordinated circuit) innovation, we attempted to delineate on FPGA (field programmable entryway exhibit) to watch the relative execution. Typically, on FPGAs, the multiplication tasks are registered in advanced flag preparing (DSP) squares, which are application explicit frameworks on chip (SoC). Along these lines, we don't hope to locate a superior multiplication technique with universally useful rationale. All multipliers are intended for marked two's supplement numbers. Both genuine and fanciful parts of the intricate numbers are 16-bit settled point numbers. Besides, we actualize the Booth encoded radix-4 and nonrecoded radix-8 multipliers for 18-bit and 24-bit contributions to think about the impact of information width to the execution just as asset use of the multipliers.

We execute the multipliers in C++ utilizing abnormal state blend (HLS) apparatus to accomplish better reusability of the code. A few advantages and hindrances of utilizing a HLS approach are talked about in this work. All multipliers are additionally checked in reproductions and orchestrated for a similar innovation target. For reference, we executed reference structures of radix-8 nonrecoded and DSP arrangements straightforwardly in VHDL also.

2. RELATED WORK

Early multiplier plans, for example, bi-area, Baugh Wooley and Hwang [2] propose the usage of a 2's supplement engineering, utilizing dull modules with uniform interconnection designs. In any case, a portion of these plans don't allow a proficient VLSI acknowledgment because of the unpredictable tree-cluster structure utilized. Progressively ordinary and reasonable multiplier structures dependent on the Booth recoding systems have been proposed [4], [5]. In the Modified Booth algorithm roughly 50% of the halfway items that should be included is utilized. In the work introduced in [1], the enhancement in postponement and power has a similar important source with respect to the Booth engineering, the decrease of the halfway item terms, while keeping the consistency of a cluster multiplier. As saw in [1], the cluster multiplier is more proficient than the Modified

Booth because of the lower rationale profundity that diminishes the measure of glitching along the circuit.

In spite of the fact that the Booth algorithm gives straightforwardness, it is now and then hard to plan for higher radices because of the multifaceted nature to pre-register an expanding number of products of the multiplicand inside the multiplier unit. In [1] it is demonstrated that the cluster multiplier can be all the more normally reached out for higher radices, utilizing less rationale levels and henceforth displaying significantly less deceptive advances. In our work we have proposed three new plans for the committed radix-16 multiplication square so as to enhance the proficiency of the exhibit multiplier of [1].

3. MODIFIED BOOTH ALGORITHM

Booth multiplier:

Regular cluster multipliers, similar to the Braun multiplier and Baugh Woolley multiplier accomplish nearly great execution yet they require huge zone of silicon, not at all like the include move algorithms, which require less equipment and show less fortunate execution. The Booth multiplier makes the utilization of Booth encoding algorithm so as to decrease the quantity of halfway items by thinking about two bits of the multiplier at once, in this manner accomplishing a speed advantage over other multiplier structures. This algorithm is legitimate for both marked and unsigned number. It acknowledges the number in two's compliment structure, in light of radix-2 calculation. It can deal with marked double multiplication by utilizing 2's compliment portrayal. This builds the multifaceted nature of how indications of the operands get put away in helper circuits.

Non-Booth Recoding:

Utilizing the non-Booth encoding technique for incomplete item age, the multiplier bits are analyzed successively beginning from LSB to MSB. On the off chance that the multiplier bit is one, the incomplete item is just the multiplicand. Something else, the halfway item is zero. Each new incomplete item is moved one piece position to one side. Every fractional item can be created by simply utilizing a line of two-information AND entryways. The quantity of halfway items produced equivalents the span of the multiplier bits.

BOOTH ALGORITHM:

Booth Recoding:

Booth's multiplication algorithm is a multiplication algorithm that increases two marked double numbers in two's supplement documentation. The algorithm was created by Andrew Donald Booth in 1951 while doing research on crystallography at Birkbeck College in Bloomsbury, London. Booth utilized work area mini-computers that were quicker at moving than adding and made the algorithm to build their speed. Booth's algorithm is of enthusiasm for the investigation of PC engineering. The Booth recoding, or Booth algorithm, was proposed by Andrew D. Booth in 1951[7]. This technique can be utilized to increase two's supplement number without the sign piece expansion. Booth recoding is a strategy for decreasing the quantity of halfway items to be summed. Booth saw that when strings of '1' bits happen in the multiplicand the quantity of incomplete items can be diminished by utilizing subtraction. The task of Booth recoding comprises of two noteworthy advances [8]: the first is to take one piece of the multiplier, and afterward to choose whether to add the multiplicand as indicated by the present and past bits of the multiplier. This encoding plan is sequential, which implies that the distinctive estimation of the 2-bits (current and past bits) compares to the diverse activities.

X_i	X_{i-1}	Operations	Comments	Y_i
0	0	Shift only	String of zeros	0
1	1	Shift only	String of zeros	0
1	0	Subtract and shift	Beginning of string of ones	1
0	1	Add and Shift	End of string of ones	1

Table 1 : Recoding in Booth Algorithm

The sequential recoding plan is normally connected in sequential multipliers. The upside of this technique is that the halfway item circuit is basic and simple to execute. Accordingly, this plan is appropriate for the execution of little multipliers. The downside is that the strategy can't proficiently deal with the sign expansion and it creates various incomplete items the same number of as the quantity of bits of the multiplier, which results in numerous adders required so the region and power utilization increment. This technique isn't appropriate for substantial multipliers.

MODIFIED BOOTH ALGORITHM:

The changed Booth encoding (MBE), or adjusted Booth's algorithm (MBA), was proposed by O. L. Macsorley in 1961 [11]. The recoding technique is broadly used to create the incomplete items for execution of huge parallel multipliers, which receives the parallel encoding plan. One of the arrangements of acknowledging rapid multipliers is to upgrade parallelism which diminishes the quantity of ensuing figuring stages. The first form of the Booth algorithm (Radix-2) had two disadvantages. They are:

(I) The quantity of include subtract tasks and the quantity of move activities winds up factor and ends up badly arranged in structuring parallel multipliers.

(ii)The algorithm winds up wasteful when there are confined 1's.

These issues are overwhelmed by utilizing adjusted Radix4 Booth algorithm .This module recodes the 16-bit multiplier utilizing radix 4 Booth's algorithm. Radix 4 encoding decreases the all out number of multiplier digits by a factor of two, which implies for this situation the quantity of multiplier digits will lessen from 16 to 8. This algorithm bunches the first multiplier into gatherings of three back to back digits where the peripheral digit in each gathering is imparted to the furthest digit of the contiguous gathering. Every one of these gatherings of three paired digits at that point compares to one of the numbers from the set {2, 1, 0, - 1, - 2}. Each recoder produces a 3-bit yield where the main piece speaks to the number 1 and 13 the second piece speaks to the number 2

Multiplier Bits			Recoded operation on multiplicand,
Y_{2i-1}	Y_{2i}	Y_{2i+1}	X
0	0	0	0X
0	0	1	+X
1	1	0	-X
0	1	1	+2X
1	0	0	-2X

Table 2 : Radix-4 Booth Algorithm

The third and last piece demonstrates whether the number in the first or second piece is negative. Since there are 16 input bits, there will be a sum of 8 Booth recoder modules in the general multiplier design. The manner in which the yields are resolved is appeared table above.

4. RADIX-4 MULTIPLIER

Booth algorithm is an incredible algorithm [5] for marked number multiplication, which treats both positive and negative numbers consistently. Since a k-bit twofold number can be translated as k/2-digit Radix-4 number, a k/3-digit Radix-8 number, etc, it can manage more than one piece of the multiplier in each cycle by utilizing high radix multiplication[6]. The real disservice of the Radix-2 algorithm was that the procedure required n shifts and a normal of n/2 increases for a n bit multiplier. This variable number of move and include activities is badly arranged for planning parallel multipliers. Likewise the Radix-2 algorithm winds up wasteful when there are disengaged 1's. The Radix-4 altered Booth algorithm beats every one of these restrictions of Radix-2 algorithm. For operands equivalent to or more noteworthy than 16 bits, the adjusted Radix-4 Booth algorithm has been broadly utilized. It depends on encoding the two's supplement multiplier so as to lessen the quantity of halfway items to be added to n/2

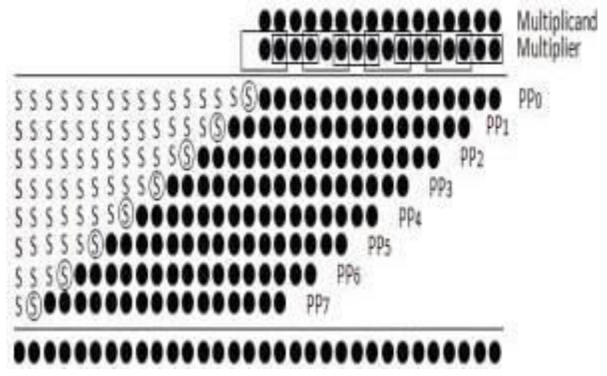

Fig 1: Radix-4 Modified Booth Algorithm

5. RADIX-8 MULTIPLIER

Radix-8 Booth Encoding multiplier utilizes 4-bit encoding plan [9] to deliver 33% the quantity of halfway items. Radix-8 Booth recoding applies indistinguishable algorithm from that of Radix-4, however at this point we take groups of four of bits rather than triplets. Every group of four is systematized as a marked digit. Radix-8 algorithm decreases the quantity of incomplete items to n/3, where n is the quantity of multiplier bits.

Multiplier Bits				*Recoded operation Multiplicand,*
Y_{i+2}	Y_{i+1}	Y_i	Y_{i-1}	*X*
0	0	0	0	0X
0	0	0	1	+X
0	0	1	0	+X
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	-X
1	1	1	0	-X
1	1	1	1	0X

Table 3 :Radix-8 modified booth algorithm

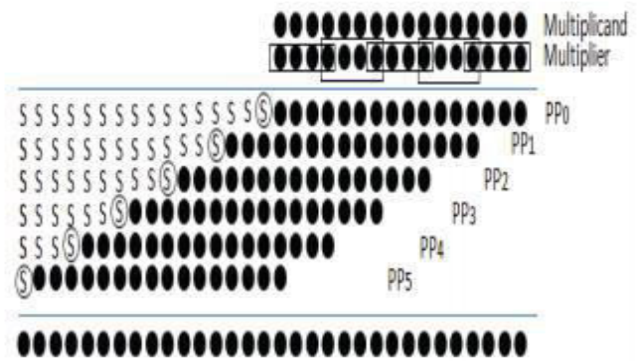

Fig 2: Example for radix-8 booth algorithm

6. CONCLUSION

Multiplication frequently restrains the execution of advanced flag handling applications. Consequently, equipment multipliers are utilized to quicken these sort of tasks. In this work, radix-4 Booth encoded, radix-8 Booth encoded, radix-8 nonrecoded complex number multipliers have been structured and actualized on FPGA. We planned radix-4 and radix-8 multiplier utilizing Modified Booth Algorithm. The radix-8 Modified Booth Multiplier has elite than the radix-4 Modified Booth Multiplier. Since, radix-8 has less number of fractional item than radix-4. Along these lines, the timeframe was diminished in radix-8 than the radix-4 multiplier. Accordingly, the proposed multipliers are relevant to High Speed information Transmission.

REFERENCES

[1] E. Costa, J. Monteiro, and S. Bampi. A New Architecture for Signed Radix 2m Pure Array Multipliers. In IEEE International Conference on Computer Design, pages 112-117, 2002

[2] K. Hwang. Computer Arithmetic – Principles, Architecture and Design. School of Electrical Engineering, 1979.

[3] E. Sentovich. SIS: a System for Sequential Circuit Synthesis. Berkeley: University of California, 1992.

[4] B. Cherkauer and E. Friedman. A Hybrid Radix-4/Radix-8 Low Power, High Speed Multiplier Architecture for Wide Bit Widths. In IEEE ISCAS, vol. 4, pages 53-56, 1996.

[5] P. Seidel, L. Mc Fearing, and D. Matula. Binary Multiplication Radix-32 and Radix-256. In 15th Symp. On Computer Arithmetic, pages 23-32, 2001.

[6] Dadda, L. Some schemes for parallel multipliers. Alta frequenza, 1965, vol. 34, no. 5, p. 349–356.

[7] Baugh, C. R., and Wooley, B. A. A two's complement parallel array multiplication algorithm. IEEE Transactions on Computers, 1973, vol. 100, no. 12, p. 1045–1047.

[8] Sam, H., and Gupta, A. A generalized multi bit recoding of two's complement binary numbers and its proof with application in multiplier implementations. IEEE Transactions on Computers, 1990, vol. 39, no. 8, p. 1006–1015.

[9] Bekiaris, D., Pekmestzi, K., and Papachristou, C. A high-speed radix-4 multiplexer-based array multiplier. Proceedings of the 18th ACM Great Lakes symposium on VLSI, 2008, p. 115–118.

[10] Basha, S. S., and Badashah, S. J. Design and implementation of radix-4 based high speed multiplier for ALU's using minimal partial products. International Journal of Advances in Engineering and Technology, 2012, vol. 4, no. 1, p. 314–325.

[11] Lin, H., Chang, R. C., and Chan, M. Design of a novel radix-4 booth multiplier. IEEE Asia-Pacific Conference on Circuits and Systems, 2004, vol. 2, p. 837–840.

[12] Lamberti, F., Andrikos, N., Antelo, E., and Montuschi, P. Reducing the computation time in (short bit-width) two's complement multipliers. IEEE transactions on computers, 2011, vol. 60, no. 2, p. 148–156.

[13]. Cooper, A. Parallel architecture modified Booth multiplier. in IEE Proceedings G-Electronic Circuits and Systems. 1988. IET.

[14]. Shanbhag, N.R. and P. Juneja, Parallel implementation of a 4* 4-bit multiplier using a modified Booth's algorithm. IEEE Journal of solid-state circuits, 1988. 23(4): p. 1010-1013.

[15]. Goto, G., et al., A 54* 54-b regularly structured tree multiplier. IEEE Journal of solid-state circuits, 1992. 27(9): p. 1229-1236.

[16]. Fadavi Ardekani, J., M* N Booth encoded multiplier generator using optimized Wallace trees. IEEE Transactions on very large scale integration (vlsi) systems, 1993. 1(2): p. 120-125.