ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

High Speed And Area Efficient Multiplier

[1]P.S.Tulasiram, [2]D. Vaithiyanathan and [3]Dr. R. Seshasayanan

[1]*PG Student, College of Engineering Guindy, Department of Electronics and Communication, Anna University, Chennai, Tamilnadu, India-6000025.*
[2]*Teaching Fellow,College of Engineering Guindy, Department of Electronics and Communication, Anna University, Chennai, Tamilnadu, India-6000025.*
[3] *Associate Professor,College of Engineering Guindy, Department of Electronics and Communication, Anna University, Chennai, Tamilnadu, India-6000025.*

ARTICLE INFO

Article history:
Received 10 March 2015
Received in revised form 20 March 2015
Accepted 25 March 2015
Available online 10 April 2015

Keywords:
Multiplier and accumulator, carry-lookahead adder, Recoding, field programmable gate array

ABSTRACT

The advancement in real-time implementation of many digital signal processing (DSP) algorithms and multimedia applications made their performance limited to the speed, energy efficiency, and area requirement of multiplication. Parallel multiplier - and accumulator (MAC) is frequently used in Digital signal processing and video/graphics applications. A new architecture of MAC for high speed arithmetic by combining multiplication with accumulation and devising a carry-lookahead adder (CLA), the performance is improved. This paper describes implementation of radix-4 Multiplier and it is compared with radix-2 and radix-8 multiplier. Radix-4 Multiplier employs both addition and subtraction and they treat both positive and negative numbers uniformly. There is no special action required for the negative numbers. In this paper, we investigate the method of implementing the Parallel MAC with minimum delay. An efficient VerilogHDL code has been written and successfully simulated and synthesized for Xilinx FPGA vertex-6 low power (Xc6vlx75tl-1Lff484) device, using Xilinx 12.2 ISE and XST. Cadence Encounter® tool using TSMC 180nm CMOS cell library is been used to elaborate and synthesis the model. The analyses obtained from implementation show that architecture is 41% faster than the Wallace tree architecture with optimal area utilization.

ToCite ThisArticle: P.S.Tulasiram, D.Vaithiyanathan and Dr. R. Seshasayanan., High speed and area efficient multiplier ***Aust. J. Basic & Appl. Sci.,*** *9(15): 22-26, 2015*

INTRODUCTION

The multiplier and multiplier-and-accumulator (MAC) (Dong-Wook Kim *et al*., 2010) are the vital elements of the digital signal processing such as convolution, filtering and inner products. Most digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) (Prasanna Raj P *et al*., 2009)or discrete wavelet transform (DWT) (Lakshmanan *et al*., 2002). Because they are basically done by continual application of multiplication and addition, the speed of the multiplication and addition arithmetic's determines the execution speed and performance of the entire system. Usually the multiplier is one of most complex task in arithmetic operation because it has to perform three basic operations i.e. partial product generation, shifting and addition, hence the multiplier requires the longest delay among the basic operational blocks in digital system. The critical path is determined more by the multiplier (Dong-Wook Kim *et al*., 2010). Furthermore, multiplier consumes much area and dissipates more power. Hence designing multipliers which offer either of the following design targets – high speed, low power consumption, less area or even a combination of them is of substantial research interest. The speed of multiplication can be optimized by reducing the number of partial productsand/or accelerating the accumulation of partial products. To reduce the number of calculation steps for the partial products, Modified Booth Algorithm (MBA) has been applied mostly where Wallace tree has taken the role of increasing the speed to add the partial products. To increase the speed of the MBA algorithm, many parallel multiplication architectures have been reported (C. S. Wallace, 1964), (J. Fadavi-Ardekani, 1993), (Rajendra Katti, 1994). Among the many methods of implementing high speed parallel multipliers, there are three basic approaches namely Radix-2, Radix-4 and Radix-8. This paper describes an efficient implementation of a high speed parallel multiplier using these approaches. Here three multipliers are proposed. The first multiplier makes

Corresponding Author: P.S.Tulasiram, PG Student, Department of Electronics and Communication Engineering,College of Engineering Guindy, Chennai, Tamilnadu, India 600025.
E-mail: tulasiramps@gmail.com.

use of the Radix-2 Algorithm and the second multiplier uses the Radix-4 Booth algorithm and third multiplier uses Radix-8 algorithm. The design is structured for N x N multiplication. The number of partial products remains same N in Radix-2 Booth algorithm while it gets reduced to N/2 in Radix-4 Booth algorithm and further to N/3 in Radix-8. This reduces the time as well as the chip area. To further enhance the speed of operation, carry-look-ahead (CLA) adder is used as the final adder.(Zimmermann *et al*, 2003) present an optimized design of (C. N. Lyu *et al*, 1995), which results in improvements in both area and critical path**.** One of the most advanced types of MAC for general-purpose digital signal processing has been proposed by(Elguibaly,2000). It is an architecture in which accumulation has been combined with the tree that compresses partial products. The architecture proposed in the critical path was reduced by eliminating the adder for accumulation and decreasing the number of input bits in the final adder. This work is extended of (Tulasiram *et al*,2014).

I. Architecture:

A. Radix-2 Model:

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 as shown in Table 1.

00 or 11 :do nothing

01:Marks the end of a string of 1s and add multiplicand to partial product

10: Marks the beginning of a string of 1s Subract multiplicand from partial product.

One of the solutions realizing high speed multiplier is to enhance parallelism which helps in decreasing the number of subsequent calculation stages. The radix-2 multiplier had two drawbacks:

1. The number of add/subtract operations became variable and hence became inconvenient while designing parallel multipliers.
2. The algorithm becomes inefficient when there are isolated 1s.

These problems are overcome by using Radix 4 algorithm which can scan strings of three.

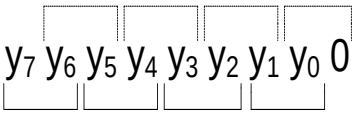

Fig. 1: Radix-2 group formation.

Table 1: Radix-2 encoding.

S.No	X_i	X_{i+1}	Y_i	Partial Product
1	0	0	0	$M \times 0$
2	0	1	1	$M \times 1$
3	1	0	-1	$M \times -1$
4	1	1	0	$M \times 0$

B. Radix-4 Model:

It is possible to reduce the number of partial products by half, by using the technique of radix 4 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 $\pm 1, \pm 2$ or 0, to obtain the same results. Radix-4 algorithm 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 the LSB, and the first block only uses two bits of the multiplier and assumes a zero for the third bit.

The functional operation of Radix-4 booth encoder is shown in the Table 2. It consists of eight different types of states and during these states we can obtain the outcomes, which are multiplication of multiplicand with 0,-1 and -2 consecutively.

Radix-4 algorithm is given below:

Step 1: Extend the sign bit 1 position if necessary to ensure that n is even.

Step 2: Append a 0 to the right of the LSB of the multiplier.

Step 3: According to the value of each vector, each partial product will be 0, +y, –y, +2y or –2y.

The negative values of y are made by taking the 2's complement and Carry-look-ahead (CLA) fast adders are used for addition in this paper. The multiplication of y is done by shifting y by one bit to the left. Thus, in any case, designing n-bit parallel multipliers only n/2 partial products are generated. The advantage of this method is the halving of the number of partial products. This is important in circuit design as it relates to the propagationdelay in the running of the circuit, and the complexity and power consumption of its implementation.

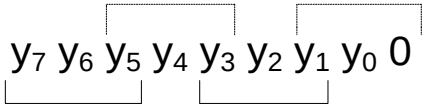

Fig. 2: Radix-4 group formation.

Table 2: Radix-4 encoding.

S.No	X_i	X_{i-1}	X_{i-2}	Y_i	Partial Product
1	0	0	0	0	$M \times 0$
2	0	0	1	1	$M \times 1$
3	0	1	0	1	$M \times 1$
4	0	1	1	2	$M \times 2$
5	1	0	0	-2	$M \times -2$
6	1	0	1	-1	$M \times -1$
7	1	1	0	-1	$M \times -1$
8	1	1	1	0	$M \times 0$

C. Radix-8 Model:

It is possible to reduce the number of partial products further to one third, by using the technique of radix 8 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$\pm 1, \pm 2, \pm 3$ or 0,to obtain the same results. Radix-8 algorithm performs the process of encoding the multiplicand based on multiplier bits. It will compare 4 bits at a time with overlapping technique. Grouping starts from the LSB, and the first block only uses three bits of the multiplier and assumes a zero for the fourth bit. The functional operation of Radix-8 booth encoder is shown in the Table 3. It consists of sixteen different types of states and during these states we can obtain the outcomes, which are multiplication of multiplicand with 0,-1,-2,-3 consecutively. Radix-8 algorithm is given below:

Step 1: Extend the sign bit 1 position if necessary to ensure that n is even.
Step 2: Append a 0 to the right of the LSB of the multiplier.
Step 3: According to the value of each vector, each Partial Product will be 0, +y, –y, +2y, –2y, +3y and -3y.

The negative values of y are made by taking the 2's complement and Carry-look-ahead (CLA) fast adders are used for addition in this paper. The multiplication of y is done by shifting y by one bit to the left. The advantage of this method is the number of partial products is reduced to one third. This is important in circuit design as it relates to the propagation delay in the running of the circuit, and the complexity and power consumption of its implementation. However the Radix-8 algorithm doesn't have anyoptimization in area and power. This is due to multiply by 3 which is done by shifting followed by CLA adder.

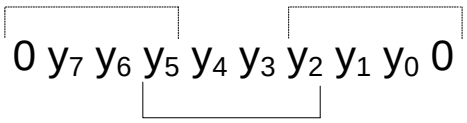

Fig. 3: Radix-8 group formation.

Table 3: Radix-8 encoding.

S.No	X_i	X_{i+1}	X_{i+2}	X_{i+3}	Y_i	Partial Product
1	0	0	0	0	0	$M \times 0$
2	0	0	0	1	1	$M \times 1$
3	0	0	1	0	1	$M \times 1$
4	0	0	1	1	2	$M \times 2$
5	0	1	0	0	2	$M \times -2$
6	0	1	0	1	3	$M \times 3$
7	0	1	1	0	3	$M \times 3$
8	0	1	1	1	4	$M \times 4$
9	1	0	0	0	-4	$M \times -4$
10	1	0	0	1	-3	$M \times -3$
11	1	0	1	0	-3	$M \times -3$
12	1	0	1	1	-2	$M \times -2$
13	1	1	0	0	-2	$M \times -2$
14	1	1	0	1	-1	$M \times -1$
15	1	1	1	0	-1	$M \times -1$
16	1	1	1	1	0	$M \times 0$

II. Experimental Results And Analysis:

In Table 4 shows the detailed report of the analysis done with radix based multiplier unit, which is implemented in Xilinx 6VLX240TFF784-3 device and simulated and synthesized using Xilinx 12.2 tool. The results were taken for radix-2, radix-4, radix-8 multiplier designed for 8 bit word. The radix-4 multiplier is optimized according to speed, generating partial products and sums. Upon comparison of the speed, we can conclude that radix-

8 recoded multiplier proves to be a better option over conventional multipliers used in several complex VLSI circuits. Though radix-8 multiplier has high speed over radix-4 but consumes more power and area. The reason is Radix-4 will perform multiply by two operation as shifting only whereas Radix-8 perform multiply by three operation which requires shifting followed by addition(carry look ahead adder). The designed radixbased multiplier also experimented in cell level to ensure its performance by using Cadence Encounter® tool using TSMC 180nm CMOS cell library. Table 5 gives optimum power consumption and area utilization within cell.

Table 4: Multiplier Unit Comparison.

Radix Type	No. of Slices Out of 150720	Delay(ns)
Wallace tree	92	10.789
Radix-2	293	9.289
Radix-4	179	6.442
Radix-8	273	6.134

Table 5: Cell Level Analysis Report.

Parameters	Wallace Tree	Radix-2	Radix-4	Radix-8
Area(um^2)	1057.69	2378.58	1382.98	1810.57
Leakage Power(nW)	3377.28	9585.07	4967.73	5803.26
Internal Power(nW)	22470.15	57958.89	29013.11	31548.16
Net Power(nW)	10932.68	16825.22	10810.76	14884.59
Switching Power(nW)	33402.83	74784.11	39824.54	46432.76
Cell Usage	203	314	215	348

The performance factors are analyzed based on the performance of the multiplier by implementing along with the standard benchmark circuits such as Matrix Multiplier (MM3) and 4-tap FIR filter (FIR4). Furthermore the cell based performance of the design is also discussed using the Cadence Encounter(R) tool, which gives the cell area and power consumption of the design While going for the FPGA embedding and island style options, this gives the portrait of the design and foresees the future enhancements.

Table 6: Matrix multiplication comparison.

Radix Type	No. of Slices Out of 150720	Delay(ns)
Radix-2	8325	14.843
Radix-4	5193	11.420
Radix-8	8010	11.064

Table 6&7 shows Power, Area and Delay analysis for matrix 3×3 multiplication using Radix-2, Radix-4 and Radix-8 Multiplier. Similarly

Table 8 & 9 shows Power, Area and Delay analysis for 4-Tap FIR Filter using Radix-2, Radix-4 and Radix-8 Multiplier.

Table 7: Cell level analysis report for matrix multiplication.

Parameters	Radix-2	Radix-4	Radix-8
Area(um^2)	68781.18	41080.74	53444.97
Leakage Power(nW)	272900.50	143183.34	171505.35
Internal Power(nW)	1925880.59	1671161.53	1174882.70
Net Power(nW)	618253.07	454589.40	574952.03
Switching Power(nW)	2544133.65	1525690.93	1749834.73
Cell Usage	9324	6678	10242

Table 8: 4-Tap Fir Filter Comparison.

Radix Type	No. of Slices Out of 150720	Delay(ns)
Radix-2	17988	24.434
Radix-4	11425	22.204
Radix-8	17262	21.640

Table 9: Cell Level Analysis Report For 4-Tap Fir Filter.

Parameters	Radix-2	Radix-4	Radix-8
Area(um^2)	158196.71	94485.70	122923.43
Leakage Power(nW)	627672.25	329321.68	394462.31
Internal Power(nW)	4622113.41	4010787.67	2819718.48
Net Power(nW)	1483807.37	1091014.56	1379884.87
Switching Power(nW)	5342680.67	3203950.95	3674652.93
Cell Usage	24242	17362	26630

III. Conclusion:

This paper delivers the design and implementation of three high performance parallel multipliers. The first multiplier makes use of the Radix-2 algorithm, the second multiplier uses the Radix-4 and the third multiplier uses the Radix-8. All the designs were implemented on vertex-6 FPGA. The multiplier using radix- 4 algorithm shows both

optimum power consumption and optimum delay when compared to other methods of multiplier. I.e. Radix-4 algorithm method is 40.2 % faster than the Wallace tree method. The reason for speed improvement is reduction in partial products i.e. the number of partial product produced by radix-4 method is$\left\lfloor\frac{N}{2}\right\rfloor$. Though radix-8 have less partial product i.e. $\left\lfloor\frac{N}{3}\right\rfloor$than radix-4.hence it has less delay put requires more area and power this is due to multiplication by 3 which requires addition followed by adder.

REFERENCES

Booth, A.D., 1952. "A signed binary multiplication technique," Quart. J.Math., IV: 236–240.

Lyu, C.N. and D.W. Matula, 1995. "Redundant binary Booth recoding," in Proc 12th Symp. ComputArithmetic, 50–57.

Wallace, C.S., 1964. "A Suggestion for a Fast Multiplier", Electronic Computers, IEEE Transactions, 13, Page(s): 14-17.

Dong-Wook Kim, Young-Ho Seo, 2010. "A New VLSI Architecture of Parallel Multiplier-Accumulator based on Radix-2 Modified Booth Algorithm", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, 18: 201-208.

Elguibaly, F., 2000."A fast parallel multiplier–accumulator using the modified Booth algorithm," IEEE Trans. Circuits Syst., 27(9): 902–908.

Goto, G., T. Sato, M. Nakajima and T. Sukemura, 1992."A 54 54 regular structured tree multiplier," IEEE J. Solid-State Circuits, 27(9): 1229–1236.

Hussin, R., 2008 ."An Efficient Modified Booth Multiplier Architecture", IEEE International Conference, pp.:1-4.

Jan, M., Rabaey, 1995. "Digital Integrated Circuits, A Design Perspective", Prentice Hall.

Fadavi-Ardekani, J., 1993. "M N Booth encoded multiplier generator using op timized Wallace trees," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 1(2): 120–125.

Lakshmanan, Masuri Othman and Mohamad Alauddin Mohd.Ali, 2002. "High Performance Parallel Multiplier using Wallace-Booth Algorithm", Semiconductor Electronics, IEEE International Conference, pp: 433- 436.

Louis, P., Rubinfield, 1975. "A Proof of the Modified Booth's Algorithm for Multiplication", Computers, IEEE Transactions, 24: 1014-1015.

Ohkubo, N., M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki and Y. Nakagome, 1995. "A 4.4 ns CMOS 54 54 multiplier using pass-transistor multiplexer," IEEE J. Solid-State Circuits, 30(3): 251–257.

Prasanna Raj, P., Rao, Ravi, 2009. "VLSI Design and Analysis of Multipliers for Low Power", Intelligent Information Hiding and Multimedia Signal Processing, Fifth International Conference, pp: 1354-1357.

Tulasiram, P.S., D. Vaithiyanathan, Dr.R. Seshasayanan, 2014. "Implementation of modified booth recoded Wallace tree multiplier for fast arithmetic circuits "International Journal of Advanced Research in Computer Science and Software Engineering, 4: 798-802.

Rajendra Katti, 1994. "A Modified Booth Algorithm for High Radix Fixedpoint Multiplication", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol. 2, pp.: 522-524.

Zimmermann, R. and D.Q. Tran, 2003 ."Optimized synthesis of sum-of-products," in Proc. Asilomar Conf. Signals, Syst. Comput., Pacific Grove,Washington, DC, 867–872.