Indian Journal of Science and Technology, Vol 8(19), DOI: 10.17485/ijst/2015/v8i19/77052, August 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Design of Fused Add-Multiply Operator using Modified Booth Recoder for Fast Arithmetic Circuits

Jonnalagadda Raghavendra* **and T. Vigneswaran**

School of Electronics Engineering, VIT University, Chennai - 600127, Tamil Nadu, India;
jonnalagadda.raghavendra2013@vit.ac.in, vigneswaran.t@vit.ac.in

Abstract

Background: In Digital Signal Processing (DSP) the complex arithmetic instructions are mostly used. The decoding of these instructions usually takes more time in many applications. **Methods**: The objective of this research work mainly focused on the delay reduction by decreasing the partial products with the help of higher radix booth recoder. The booth recoder plays a key role in fused add-multiply operation for partial product generation. **Findings:** The proposed fused add multiply unit reduces the delay by reducing the number of partial products which is very useful for fast arithmetic circuits. The fused add-multiply units are simulated in Xilinx® 14.3 ISE in Virtex-5 environment and synthesized in Cadence® RTL Compiler and layout generated from Cadence® Encounter in 180nm technology. **Conclusion:** Based on experimental results, it is observed that the delay of the proposed method is reduced by 17.5% than the existing work.

Keywords: Accumulate Units, Booth Recorder, Fast Arithmetic, Fused Add Multiply, Multiply

1. Introduction

Now-a-days Digital Signal Processing is an essential part of consumer electronics providing custom accelerators for the domains of communications and multimedia etc[1]. The major DSP applications involve in a huge number of arithmetic operations as their implementation is based on computationally intensive kernels, such as Fast Fourier Transform (FFT), Finite Impulse Response (FIR) filters, Discrete Cosine Transform (DCT) and signal convolution. The DSP System's performance depends upon the design of the arithmetic architecture design. The research in the area of arithmetic optimization[2, 3] specifies that the combining the arithmetic components like addition and multiplication as a single unit can lead to great improvement in terms of performance. The observation on an addition that is subsequent to multiplication like in symmetric FIR filters, the Multiply-Accumulator (MAC) and Multiply-Add (MAD) units were introduced in[4, 10] leading to more efficient implementation of DSP algorithms compared to the conventional ones, which use only primitive resources[5]. There are many DSP applications that are based on Fused Add-Multiply (FAM) operation for the instance FFT algorithm.

The add-multiply operator shown in[6] uses the Modified Booth recorder which uses Radix-4 recoding cannot be used for faster arithmetic circuits as the partial products reduces by only N/2, whereas in the proposed work for faster arithmetic circuits we can opt for higher radix recoding like Radix-8[7, 11] which reduces the number of partial products by N/3. The second proposed method is to use Radix-16 Booth-4 recoding[8, 11] which reduces the number of partial products by N/4. The third proposed method is to use Radix-32 Booth-5 recoding[9, 11] which reduces the number of partial products by N/5. By using all the proposed methods significant reduction in terms of delay can be achieved. The performance is evaluated by comparing the proposed methods with one another and the existing ones.

*Author for correspondence

The rest of the paper is organized is as follows: In section 2, the fused add-multiply operation is reviewed. In section 3, the proposed radix structures are discussed. In section 4, experimental evaluation and analysis is discussed. In section 5, results are compared and analyzed. In section 6, the conclusion of the entire proposed work.

2. FAM Implementation

In this paper, the focus is on FAM units shown in Figure 1 which develops the operation P = X (A+B). The existing Add-Multiply operation is done by using radix-4 modified booth recoding. The major drawback of this technique is the more number of partial products. The proposed methods are very good at reducing the number of partial products. As a result, the critical path delay can be reduced. In this work, we present the three types of recoding schemes for partial product reduction, i.e. Radix-8, Radix-16, Radix-32 Modified Booth Recoding.

2.1 Overview of Modified Booth Recoding

Modified Booth recoding is mostly used to recode scheme because of its advantages like less area and less power consumption. It reduces the number of partial products by half of its bit length.

Let us consider the multiplication of the 2's complement numbers X and Y with each number of having of n = 2k bits. You can be represented in the Modified Booth form as:

$$Y = \langle y_{n-1} y_{n-2} \cdots y_1 y_0 \rangle_{2's} = -y_{2k-1}.2^{2k-1} + \sum_{i=0}^{2k-2} y_i.2^i \quad (1)$$

$$y_j^{MB} = -2y_{2j+1} + y_{2j} + y_{2j-1} \quad (2)$$

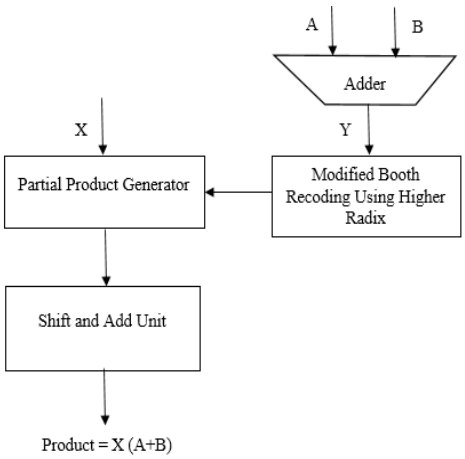

Figure 1. Fused add-multiply operator.

Where, $0 \le j \le k-1$, implies to the three consecutive bits y_{2j+1}, y_{2j-1} and y_{2j-1} with one bit overlapped and assuming that $y_{-1} = 0$. Based on the Table 1 shown below the bits need to be recorded and the partial products must be generated.

3. Proposed Higher Radix Schemes

3.1 Radix-8 Booth-3 Recoding Scheme

In radix-8 recoding is done by grouping four bits at a time shown in Figure 2 and it will result in recoded bit, which corresponds to partial product generation.

The extra operations other than Radix-4 recoding are ±3X Multiplicand, ±4X Multiplicand can be done by the left shift once and add it again and left shift twice. The operations to be performed for partial product generation are as shown in Table 2.

3.2 Radix-16 Booth-4 Recoding Scheme

In the Radix-16 Booth-4 recoding scheme five bits are taken at a time for recording and the resulted recoded bit will be used for partial product generation. The recoding of Radix-16 are {0, ±1, ±2, ±3, ±4, ±5, ±6, ±7, ±8}. The additional operations apart from the Radix-8 recoding are ±5, ±6, ±7, ±8. The operation ±5x Multiplicand can be

Table 1. Radix-4 ooth-2 recoding table

Multiplier Bits	Recoded Digit	Operation to be performed
000	0	0XM
001	+1	1XM
010	+1	1XM
011	+2	2XM
100	–2	–2XM
101	–1	–1XM
110	–1	–1XM
111	0	0XM

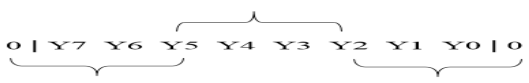

Figure 2. Radix-8 booth-3 recoding scheme.

Table 2. Radix-4 Booth-3 recoding table

Multiplier Bits	Recoded Digit	Operation
0000	0	0XM
0001	+1	+1XM
0010	+1	+1XM
0011	+2	+2XM
0100	+2	+2XM
0101	+3	+3XM
0110	+3	+3XM
0111	+4	+4XM
1000	–4	–4XM
1001	–3	–3XM
1010	–3	–3XM
1011	–2	–2XM
1100	–2	–2XM
1101	–1	–1XM
1110	–1	–1XM
1111	0	0XM

done by 4X+X or 6X-X. The operation ±6x Multiplicand can be done by three times left shifting the multiplicand. The operation ±7x Multiplicand can be done by 6X+X or 8X-X. The operation ±8x Multiplicand can be done by four times left shifting the multiplicand. The complete operations for partial product generation are shown in the Table 3.

3.3 Radix-32 Booth-5 Recoding Scheme

In Radix-32 Booth-5 recoding scheme six bits are taken at a time for recoding and the resulted recoded bit will be used for partial product generation. The recoding scheme is shown in Figure 4. The simplified structure of the recoded digits of radix-32 are given below {0, ±1, ±2, ±3, ±4, ±5, ±6, ±7, ±8, ±9, ±10, ±11, ±12, ±13, ±14, ±15, ±16}. The additional operations apart from the Radix-16 are ±9, ±10, ±11, ±12, ±13, ±14, ±15, ±16. The operation ±9x Multiplicand, can be done by

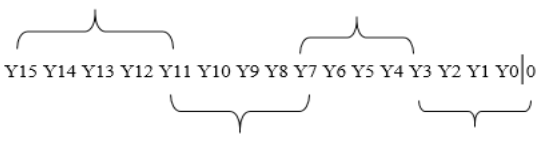

Figure 3. Radix-16 booth-4 recoding scheme.

Table 3. Radix-16 booth-4 recoding table

Multiplier Bits	Recoded Digit	Operation to be performed
00000,11111	0	0xM
00001,00010	+1	+1xM
00011,00100	+2	+2xM
00101,00110	+3	+3xM
00111,01000	+4	+4xM
01001,01010	+5	+5xM
01011,01100	+6	+6xM
01101,01110	+7	+7xM
01111	+8	+8xM
10000	–8	–8xM
10001,10010	–7	–7xM
10011,10100	–6	–6xM
10101,10110	–5	–5xM
10111,11000	–4	–4xM
11001,11010	–3	–3xM
11011,11100	–2	–2xM
11101,11110	–1	–1xM

8X+X or 10X-X. The operation ±10x Multiplicand, can be done by five times left shifting the multiplicand. The operation ±11x Multiplicand, can be done by 10X+X or 12X-X. The operation ±13x Multiplicand, can be done by six times left shifting the multiplicand. The operation ±13x Multiplicand, can be done by 12X+X or 14X-X. The operation ±14x Multiplicand can be done by seven time left shifting the multiplicand. The operation ±15x Multiplicand can be done by 14X+X or 16X-X. The operation ±16x Multiplicand can be performed by eight times left shifting the multiplicand. The complete information regarding the Booth-5 recoding is provided in the following Table 4.

4. Results and Analysis

All the FAM units are designed in IEEE 1364-2001 Verilog HDL. The Verilog HDL code is simulated and synthesized

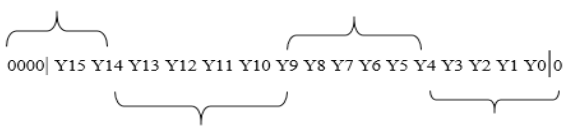

Figure 4. Radix-32 Booth-5 Recoding Scheme

Table 4. Radix-32 booth-5 recoding table

Multiplier Bits	Recoded Digit	Operation to be performed
000000,111111	0	0xM
000001,000010	+1	+1xM
000011,000100	+2	+2xM
000101,000110	+3	+3xM
000111,001000	+4	+4xM
001001,001010	+5	+5xM
001011,001100	+6	+6xM
001101,001110	+7	+7xM
001111,010000	+8	+8xM
010001,010010	+9	+9xM
010011,010100	+10	+10xM
010101,010110	+11	+11xM
010111,011000	+12	+12xM
011001,011010	+13	+13xM
011011,011100	+14	+14xM
011101,011110	+15	+15xM
011111	+16	+16xM
100000	−16	−16xM
100001,100010	−15	−15xM
100011,100100	−14	−14xM
100101,100110	−13	−13xM
100111,101000	−12	−12xM
101001,101010	−11	−11xM
101011,101100	−10	−10xM
101101,101110	−9	−9xM
101111,110000	−8	−8xM
110001,110010	−7	−7xM
110011,110100	−6	−6xM
110101,110110	−5	−5xM
110111,111000	−4	−4xM
111001,111010	−3	−3xM
111011,111100	−2	−2xM
111101,111110	−1	−1xM

in Xilinx® 14.3 ISE Virtex-5 environment and Cadence® RTL Compiler in TSMC 180nm Technology. The layout was generated in Cadence® Encounter using 180nm Technology. The delay of the FAM units gets reduced if the number of partial products is reduced. The delay got reduced by 17.5% than the existing FAM as shown in Table 5.

Table 5. Comparison between the existing and proposed

Parameter	Existing FAM [6]	Proposed FAM Radix-4 (8-bit)	Proposed FAM Radix-8(8-bit)
Delay (nSec)	6.442	5.45	5.31

The delay comparison of 16 bit and 32 bit length for radix 4,8,16 and 32 are shown in Figure 5. In Table 6(a) the design constraints like area, power and delay are compared for Radix-4,-8,-16,-32. The delay got reduced from Radix-4 to Radix-32 by 46%. Area and power got increased from the radix-4 to radix-32. In Table 6(b) the design constraints like area, power and delay are compared for radix-4,-8,-16,-32. The delay got reduced from Radix-4 to Radix-32 by 77.1%. Area and power got increased from Radix-4 to Radix-32

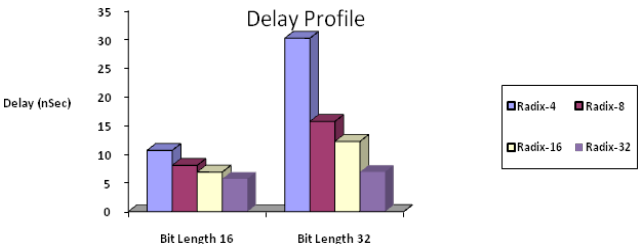

Figure 5. Delay comparison of the proposed techniques.

Table 6(a). Area, power and delay comparison

Bit Length -16			
FAM Using	Area (um²)	Power (mW)	Delay (nsec)
Radix-4	15853.62	3.042	10.686
Radix-8	27219.93	5.163	8.082
Radix-16	33270	4.743	6.904
Radix-32	49300.57	5.689	5.731

Table 6(b). Area, power and delay comparison

Bit Length -32			
FAM Using	Area (um²)	Power (mW)	Delay (nSec)
Radix-4	59848.59	16.805	30.25
Radix-8	93392.01	21.704	15.739
Radix-16	112000	19.628	12.252
Radix-32	160332.48	21.261	6.927

In the Figure 6, A, B, X are the inputs and Product is the output of the FAM unit. For instance A = 28707, B= 873 and X= –3305 gives the product as –97761900. The number of partial products reduces to 6. The critical path delay gets reduces due to partial products, so the total delay gets reduced.

In Figure 7 GDSII layout of Radix-8 FAM 16-bit length is shown. In layout generation process there is no setup and hold violations, if any violation occurs then it can be removed by optimization of the design and clock tree synthesis.

In Figure 8 A, B, X are the inputs and Product is the output of the FAM unit. For instance A = 256192145, B = 13456, X = 256911449 gives the product as 65822152194825849. The number of partial products reduces to 7. The critical path delay gets reduced due to partial product reduction, so the total delay gets reduced.

In Figure 9, GDSII layout of Radix-32 FAM 32-bit length is shown. In the layout generation process there are no setup and hold violations, if any violation occurs then it can be removed by optimization of the design and clock tree synthesis.

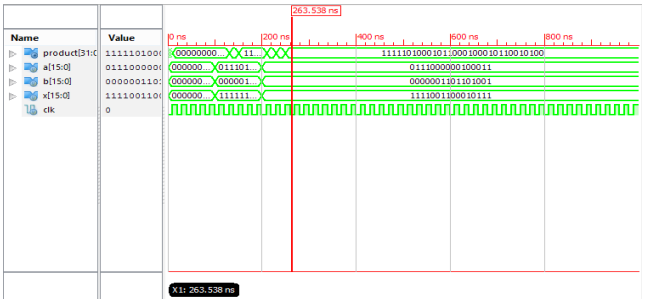

Figure 6. Simulation result of Radix-8 FAM 16-bit length.

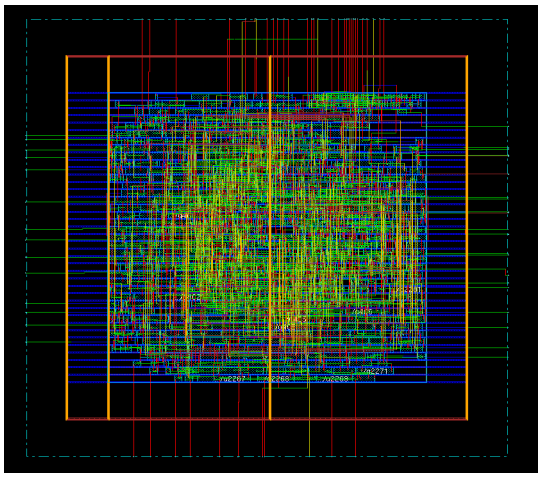

Figure 7. GDSII layout of Radix-8 FAM 16-bit length.

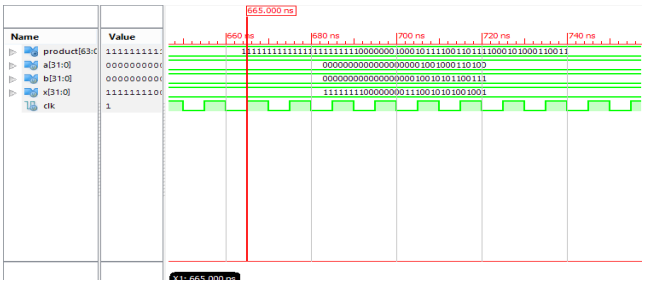

Figure 8. Simulation result of Radix-32 FAM 32-bit length.

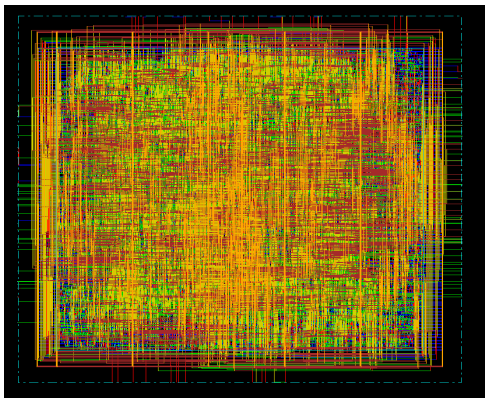

Figure 9. GDSII layout of Radix-32 FAM 32-bit length.

5. Conclusion

This paper majorly focusing on the reduction of the delay by reducing the number of partial products. The proposed methodologies are based on higher radix schemes for partial products as well as delay reduction. The delay is reduced by 17.5% than the existing work. The delay got reduced from Radix-4 to Radix-32 by 46% for proposed architecture. The delay got reduced from Radix-4 to Radix-32 by 77.1% for proposed architecture. With the obtained results the proposed FAM unit can be used in DIF- FFT Algorithm which is mainly used for reducing the computations and computation time.

6. References

1. Tsoumanis K, Xydis S, Efstathiou C, Moschopoulos N, Pekmestzi K. An optimized modified booth recoder for efficient design of add-muliply operator. IEEE Trans Circuits and Systems-I: Regular Papers. 2014 Apr; 61(4):1133–43.
2. Amaricai A, Vladutiu M, Boncalo O. Design issues and implementations for floating-point divide-add fused. IEEE Trans Circuits Syst. II-Exp. Briefs. 2010 Apr; 57(4):295–9.

3. Swartzlander EE, Saleh HHM. FFT implementation with fused floating-point operations. IEEE Trans Comput. 2012 Feb; 61(2):284–8.
4. Cavanagh JJF. Digital computer arithmetic. NewYork: McGraw-Hill; 1984.
5. Nikolaidis S, Karaolis E, Kyriakis-Bitzaros ED. Estimation of signal transition activity in FIR filters implemented by a MAC architecture. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2000 Jan; 19(1):164–9.
6. Tulasiram PS, Vaithiyanathan D, Seshasayanan R. Implemenattion of modified booth recoded wallace tree multiplier for fast arithmetic circuits. International Journal of Advanced research in Computer Science and Software Engineering. 2014 Oct; 4(10):798–802.
7. Thomas M. Design and simulation of radix-8 booth encoder multiplier for signed and unsigned numbers. International Journal for Innovative Research in Science and Technology. 2014 Jun; 1(1):1–10.
8. Bewick GW. Fast multiplication: algorithms and implementation [PhD dissertation]. Dept of Electrical Engg, Stanford University; 1994 Feb.
9. Singh JP, Kaur R, Mehta V. Delay-power performance of high speed radix 32 booth multiplier in 40nm process technology. IOSR Journal of VLSI and Signal Processing. 2014 Sep-Oct; 4(5):54–9.
10. Yeh WC. Arithmetic module design and its application to FFT [PhD dissertation]. Chiao-Tung: Dept Electron Eng, National Chiao-Tung University; 2001.
11. Swee KLS, HaiHiung L. Performance comparison review of radix-based multiplier designs. International Conference on Intelligent and Advanced Systems; Kuala Lumpur. 2012 Jun 12-14. p. 854–9.