International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 05, May-2015

Efficient Implementation of Multiplier for Digital FIR Filters

Smitha N Mallya
Dept. of EXTC
FCRIT, Vashi
Mumbai, India

Asst Prof. Sneha Revankar
Dept. of EXTC
FCRIT, Vashi
Mumbai, India

***Abstract*— This paper presents the method for implementing the multiplier for digital Finite Impulse Response (FIR) filter that requires optimized area and low power consumption. Multiplication plays a vital role in most of the high performance systems. The methods include Modified Booth Encoding Algorithm along with carry save adder, shift/add multipliers. These techniques are applied to FIR filters to minimize the area and power consumption. The proposed designs for FIR filters will be designed using Verilog HDL and synthesized, implemented using Xilinx ISE.**

***Key Terms*—FIR, Booth's Algorithm, Modified Booth's Algorithm, Xilinx ISE, FPGA.**

I. Introduction

Finite impulse response (FIR) filters are widely used in various DSP applications. In some applications, the FIR filter circuit must be able to operate at high sample rates, while in other applications, the FIR filter circuit must act as a low-power circuit that operates at moderate sample rates. The structure of the multiplier circuit also affects the resultant power consumption and speed. Choosing multipliers with more hardware breadth rather than depth would not only reduce the delay, but also the total power consumption. A lot of design methods of low power digital FIR filter have been proposed. They use a modified common sub expression elimination algorithm to reduce the number of adders used in the multiplication operation.

Multiplication is a most commonly used operation in many computing systems. In fact multiplication is nothing but addition since, multiplicand adds to itself multiplier number of times gives the multiplication value between multiplier and multiplicand. But one should consider the fact that this kind of implementation really takes many hardware resources and the circuit operates at utterly low speed. In order to address this issue so many ideas have been presented so far for the last three decades [8]. Each one is aimed at a particular improvement according to the requirement. One may be aimed at high clock speeds and another may be aimed for low power consumption or less area occupation [3], [6]. Either way ultimate job is to come up with an efficient architecture which can address three constraints of VLSI speed, area, and power. Among these three, speed is the one which requires special attention. If we observe closely multiplication operation involves two steps one is producing partial products and other is adding these partial products. Thus, the speed of a multiplier hardly depends on how fast the partial products are generated and how fast we can add them together. If the numbers of partial products to be generated are less then it is indirectly means that we have achieved the speed in generating partial products. Booth's algorithms are meant for this only. To speed up the addition operation among the partial products we need fast adder architectures. Since the multipliers have a significant impact on the performance of the entire system, many high performance algorithms and architectures have been proposed by Renuka Narasimha, Rajasekhar and Sujana Rani [1].

The next sections give a brief summary of fir filter theory and present the multiplication architecture adopted in our implementation.

II. DIGITAL FILTER THEORY

Digital filters are typically used to modify or alter the attributes of a signal in the time or frequency domain. The most common digital filter is the linear time-invariant (LTI) filter. An LTI interacts with its input signal through a process called linear convolution, denoted by y = f * x where f is the filter's impulse response, x is the input signal, and y is the convolved output. The linear convolution process is formally defined by:

$$Y[n] = x[n] * f[n] = \sum_{k=0}^{L-1} x[n]f[n-k]$$

$$= \Sigma_{k=0} f[k]x[n-k] \qquad (1)$$

LTI digital filters are generally classified as being finite impulse response (i.e., FIR), or infinite impulse response (i.e., IIR). An FIR filter is a filter whose impulse response settles to zero in finite time. An IIR filters may have an internal feedback and continue to respond indefinitely. As the name implies, an FIR filter consists of a finite number of sample values, reducing the above convolution sum to a finite sum per output sample instant. An FIR with constant coefficients is an LTI digital filter. The output of an FIR of order or length L, to an input time-series x[n], is given by a finite version of the convolution sum given in Eq (1), namely:

$$y[n]=x[n]*f[n]=\sum_{k=0}^{L-1} f[k]x[n-k] \qquad (2)$$

where $f[0] \neq 0$ through $f[L-1] \neq 0$ are the filter's L coefficients. They also correspond to the FIR's impulse response. For LTI systems it is sometimes more convenient to express in the z-domain with

$$Y(z) = F(z)\,X(z) \quad (3)$$

where F (z) is the FIR's transfer function defined in the z-domain by

$$F(z)=\sum_{k=0}^{L-1} f(z)z^{-k} \quad (4)$$

The L^{th}-order LTI FIR filter is graphically interpreted in Figure.1. It consists of a collection of a "tapped delay line," adders, and multipliers. One of the operands presented to each multiplier is an FIR filter coefficient, often referred to as a "tap weight".

The FIR filter with transposed structure Figure.1 has registers between the adders and can achieve high throughput without adding any extra pineline registers.

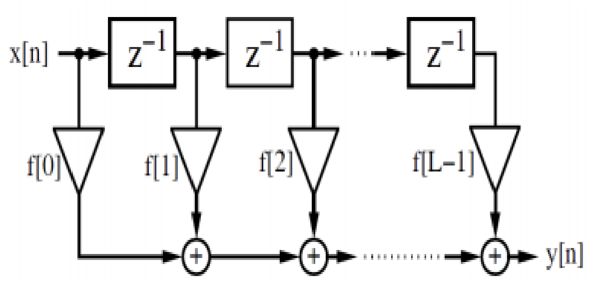

Figure 1: FIR filter in the transposed structure

.

III. FIR IMPLEMENTATION

The multiplier is one of the essential elements of the digital signal processing such as filtering, convolution, and inner products. Most digital signal processing methods use nonlinear functions such as discrete cosine transform (DCT) or discrete wavelet transform (DWT). Because they involve repetitive application of multiplication and addition, the speed of the multiplication and addition determines the execution speed and performance of the entire calculation. Because the multiplier requires the longest delay among the basic operational blocks in digital system, the critical path is determined by the multiplier, in general.

Fast multipliers are an important part of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today. In the past, multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow. It is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them. The basic multiplication principle is twofold i.e., evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The multiplier is successfully shifted and gates the appropriate bit of the multiplicand. The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix. They are then added to form the final product. For high-speed multiplication, there are some of the methods discussed in this paper.

A. Shift and Add Multiplier

In this section we present a simple Shift and Add structure for multiplier used in FIR filters [2]. It performs multiplication by generating partial products. It shifts the multiplicand left by one bit after every partial product calculation. The partial product of the current stage is set to the sum of the previous partial product and the shifted multiplicand of the current stage or 0, depending on whether the multiplier bit in the current stage is 1 or 0.

Reference Model: Shift-and-Add for 3-bit operands
Stage 1.
Rule a: product = product + mcand if(y [0])
Rule b: product = product + 0 if (! y [0])
Stage 2.
Rule a: product = product + mcand<<1 if(y [1])
Rule b: product = product + 0 if (! y [1])
Stage 3.
Rule a: product = product + mcand<<2 if(y [2])
Rule b: product = product + 0 if (! y [2]).

The same procedure is followed for n-bit multiplication. In this paper we have proposed a 16X16 bit shift and Add multiplier.

B. Modified Booth Multiplier

In order to achieve high-speed multiplication, modified Booth algorithm has been presented in this section. Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are to be multiplied [7]. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth 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 ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation.

1) Radix-2 Multiplication

The simple multiplication generator can be used to reduce the number of partial products by grouping the bits of the multiplier into pairs, and selecting the partial products from the set 0, +-M, where M is the multiplicand. Here the multiplier is grouped into two bits. Each encoded digit performs some operation on the multiplicand generating the partial product with the help of the selection Table 1. Each partial product is shifted one bit position to the left with respect to its neighbors. These partial products are then added to obtain the final product.

Table 1: Partial product selection table for radix 2

Multiplier bits	Selection
00	0
01	+Multiplicand
10	-Multiplicand
11	0

2) Radix -4 Multiplication

The Radix-4 Modified Booth's Algorithm reduces the number of partial products by about a factor of two. This selects the partial products from the set of 0, +-M, +-2M, where M is the multiplicand. The multiplier is appended by a '0' on LSB; we will call this bit as Z. The multiplier is partitioned into overlapping groups of 3 bits, and each group is decoded to select a single partial product as per the selection Table 2. Each partial product is shifted 2 bit positions with respect to its neighbors. The number of partial products will be reduced from 16 to 9 for a 16X16 multiplication. In general the there will be (n+2)/2 partial products, where n is the operand length.

Table 2: Partial product selection table for radix 4

Multiplier bits	Selection
000	0
001	+Multiplicand
010	+Multiplicand
011	+2Multiplicand
100	-2Multiplicand
101	-Multiplicand
110	-Multiplicand
111	0

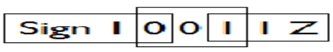

Figure 2: Recoding of multiplier

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand as illustrated in the Table 2.

The modified Booth's algorithm (*radix-4 recoding*) starts by appending a zero to the right of x_0 (multiplier LSB). Triplets are taken beginning at position x_{-1} and continuing to the MSB with one bit overlapping between adjacent triplets. If the number of bits in the multiplier (excluding x_{-1}) is odd, the sign (MSB) is extended one position to ensure that the last triplet contains 3 bits. In every step we will get a signed digit that will multiply the multiplicand to generate a partial product entering the Carry save adder.

C.Architecture for Booth Multiplication

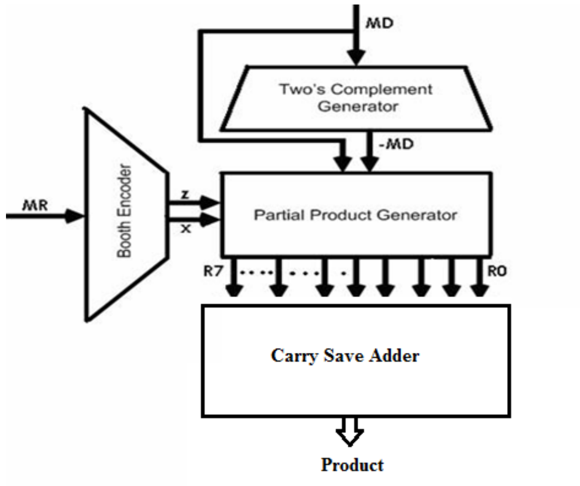

Figure 3: Architecture for Booth Multiplication

The multiplier takes two n-bit inputs: the multiplier (MR) and the multiplicand (MD), and produces the 2n-bit multiplication result of the two as its output. The architecture of the booth multiplier primarily consists of four major modules as shown in Fig.3. They are: 2's Complement Generator, Booth Encoder, Partial Product Generator and Carry Save Adder. The multiplier has been constructed in its simplest conceptual form. We will be using original Booth's Algorithm (Radix 4 encoding) for the Booth Encoder [4]. The 2's Complement Generator is used to generate the two's complement of the multiplicand. The 2's Complement Generator takes the multiplicand MD as its input and produces -MD as its output. 2's complement is required when recoded multiplier bit signifies negative multiplication. 2's complement is generated by inverting all bits of the multiplicand and then adding 1 using a ripple carry adder. The Partial Product Generator uses two control signals x and z produced by the Booth Encoder and uses these signals to choose from and extend signs of '0', MD,-MD,2MD or -2MD for creating partial products. There will be n/2 partial products if 'n' is even, (n+1)/2 partial products, if 'n' is odd. The final intermediate results are added using a Carry Save Adder. Carry Save Adder (CSA) adds two numbers with very lower latency. The carry save adder will avoid the unwanted addition and thus minimize the switching power dissipation.

IV. Simulation Results

The multiplier is designed using verilog HDL and simulated using Xilinx ISE. The Fig.4 below shows the simulation result for 16x16 Shift and Add Multiplier. Fig.5. shows the simulation result of a radix2 booth multiplication taking two 16 bit inputs.

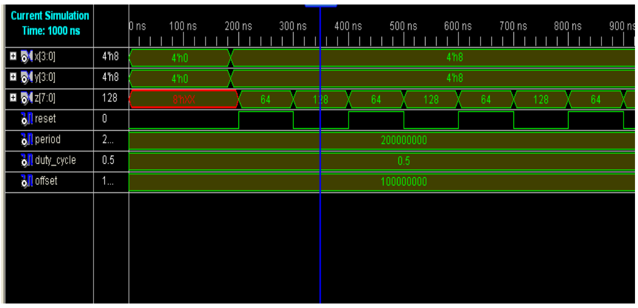

Figure 4: Simulation result of Shift and Add Method

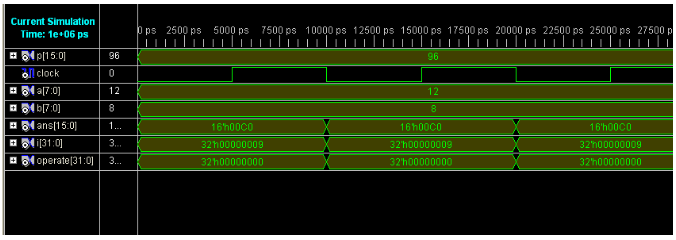

Figure 5: Simulation result for Radix2 Booth Multiplication

A. Comparison

The number of partial products in Radix 8 is less as compared to Radix4. Hence Radix 8 architecture is faster than Radix4.It is clear that Radix 8 requires less number of transistor switching and reduces power consumption.

V. CONCLUSION

In this paper we are presenting a efficient way to architecture for multiplier of FIR filter. For reducing power consumption and area we are using Modified Booth Encoding Algorithm. The above multiplication techniques presented in this paper result in the reduction of the number of partial products, as the number of partial products reduce the ime taken by adder to calculate the final product will be greatly reduced contributing in high speed multiplication. As the number of transistor switching will be less, in the total power consumption will also be reduced leading to efficient implementation of multiplier. Hence it can be concluded that Radix4 is faster compared to Radix2 and normal shift add multiplier .and thus can be used in high speed multiplication. As a future advancement in same a radix 8 encoded multiplier algorithm can be designed which will further reduce the partial products and simultaneously the power consumption.The proposed FIR filters can be designed and synthesized using Verilog in Xilinx ISE and Cadence. The design is finally implemented using Xilinx ISE Spartan 3E FPGA.

ACKNOWLEDGMENT

I would like to thank my guide Asst Prof .Sneha Revankar, of the department of Electronics and Telecommunication, FCRIT, Vashi for her constant guidance and encouragement.

REFERENCES

[1] A.Renuka Narasimha, K.Rajasekhar,A.Sujana Rani , "Implementation of low area and power efficient architectures for digital FIR filters", IJARCSSE ,Volume 2, Issue 8, August 2012.

[2] Shahnam Mirzaei, Anup Hosangadi, Ryan Kastner,"FPGA implementation of high speed FIR filters using Add and Shift method", IEEE, 2006

[3] Yun-Nan Chang,Janardhan H Sathyanarayana,Keshab K. Parhi,"Design and implementation of low power digital Serial Multipliers," University of Minnesota,Minneapolis

[4] B.N. Manjunatha Reddy, H. N. Sheshagiri, Dr. Shanthala S.,"Area Optimization of 8-bit multiplier using gate diffusion input logic," International Journal of Advanced Trends in Computer Science and Engineering,2013

[5] J.Umamaheshwari M.Veni Saranya,"Asic implementation of low power High Radix Booth Encoded Multiplier using Spst", International Journal of Communications and Engineering ,March 2012

[6] H. S. Krishnaprasad Puttam, P. Sivadurga Rao & N. V. G. Prasad,."Implementation of Low Power and High Speed Multiplier-Accumulator Using SPST Adder and Verilog ", International Journal of Modern Engineering Research ,Sept-Oct 2012

[7] Sarita Chouhan Kota, Yogesh Kumar, "Low Power Designing of FIR Filters," IJATER ,May 2012

[8] Rupali Madhukar Narsale, Dhanasri Gawali,"Design and implementation of low power FIR filter:A review" International Journal of VLSI and Embedded Systems-IJVES,March-April 2013