Vol 06, Article 09634; October 2015
http://ijves.com

International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

FIR Filter Design and Implementation using Remez Exchange Algorithm with Multipliers and Adders

Vadapalli Siddhartha[1]
[1]Electronics and Communication Engineering, Geethanjali College of Engineering and Technology, Cheeryala Village, Keesara Mandal, Rangareddy Dist., Hyderabad, Telangana, India
[1]siddhartha_vadapalli@yahoo.co.in

ABSTRACT

FIR filter can be designed by using either windowing method or by using the iterative method. Here the iterative method of Remez Exchange algorithm is used. The main aim is to design a low pass FIR filter by using various multipliers and adders. The faster execution speed and smaller memory size are the important factors in designing the DSP systems. Here in the paper the multipliers that have been considered are the Booth Multiplier, Wallace Tree Multiplier. The adders that have been considered are the Carry Skip Adder and the Carry Save Adder. The above mentioned multipliers are combined with the adders in the design of FIR filter. These proposed multipliers will do the computation by using less number of adders and steps. This criterion is very important in the fabrication of the chips and the high performance systems require components which are as small as possible. Here first the coefficients for the FIR filter using Remez Exchange algorithm have been obtained using Matlab and then the sine wave samples that are to be given as the inputs to FIR filter have also been obtained from Matlab. Then the proposed multipliers and adders have been designed using Verilog HDL. Once the adders and multipliers have been designed then their combination of each multiplier with adder in the FIR filter structure will be used. Once all the combination of multipliers and adders has been done in the design of the FIR filter then a comparison has been made to say which combination of multipliers and adders will be the best in terms of memory size and speed. In the comparison when Booth multiplier with carry save adder is used the speed is less when compared with the speed obtained with the combination of Booth multiplier with carry skip adder. Coming to the next design constraint i.e. memory size, the combination of Booth multiplier with carry save adder has got less memory size when compared with the combination of Booth multiplier with carry skip adder. Similarly for the combination of Wallace tree multiplier with carry skip adder it has got more speed when compared with the combination of Wallace tree multiplier with carry save adder. Coming to memory size the combination of Wallace tree multiplier with carry save adder has got less memory when compared with the combination of Wallace tree multiplier with carry skip adder. But between the combination of Booth multiplier and Wallace tree multiplier with adders it is been seen that Wallace multiplier with adders has got more speed and also less memory when compared with the Booth multiplier combination with adders.

Keywords: *Booth multiplier, Carry save adder, Carry skip adder, FIR filter, Iterative method, Remez Exchange algorithm, Wallace tree multiplier*

[1] INTRODUCTION

There are many ways to design a FIR filter like windowing or frequency sampling methods. These methods have the problem in control of critical frequencies. So the problem of these critical frequencies can be overcome by the optimal design method and such a method can be done by the use of an algorithm called Remez Exchange Algorithm. By using this Remez Exchange Algorithm method the values will be obtained accurately. Also the stop band and pass band will be displayed clearly with maximum number of ripples. By the use of this Remez Exchange algorithm there are many advantages that can be obtained such as design is very optimal and also simple. The method of the Remez Exchange is very flexible and can be used to design filters, differentiators, and Hilbert Transforms. These all designs cannot be obtained with the normal design of the filter using windowing or frequency sampling methods. The Remez Exchange [1], [2] [3] algorithm uses the mathematical optimization technique and is considered to be cleverer method than the window design method. The steps that have been mentioned above in the window design method can be done automatically in the Remez Exchange Method. It iterates between the filter coefficients and the actual frequency response until it finds the filter that just meets the specification with the lowest possible number of filter coefficients. Actually, the Remez Exchange algorithm never really calculates the frequency response: but it does keep comparing the actual with the design goal. The Remez method produces a filter which just meets the specification without over performing. Many of the window method designs actually perform better when moving further away from the passband. This is considered to be a wasted performance because they are using more filter coefficients than they need. Similarly, many of the window method designs actually perform better than the specification within the passband which can also be considered as a wasted performance, and also using more filter coefficients than they need. So Remez designs have equal ripple - up to the specification but no more - in both passband and

2010-2014 – IJVES

Vol 06, Article 09634; October 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

stopband. This is the reason they are often called equiripple designs. The figure below shows how the ripples will come in the passband and stopband.

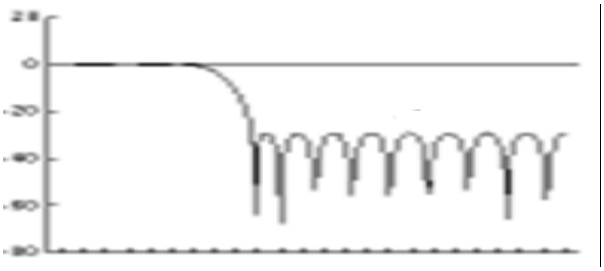


Fig. 1. Equal ripples in both passband and stopband

2. FIR FILTER

In signal processing, there are many instances in which an input signal to a system contains extra unnecessary content or additional noise which can degrade the quality of the desired portion. In such cases we may remove or filter out the useless samples. For example, in the case of the telephone system, there is no reason to transmit very high frequencies since most speech falls within the band of 400 to 3,400 Hz. Therefore, in this case, all frequencies above and below that band are filtered out. The frequency band between 400 and 3,400 Hz, which isn't filtered out, is known as the pass band, and the frequency band that is blocked out is known as the stop band. FIR, Finite Impulse Response, filters are one of the primary types of filters used in Digital Signal Processing. FIR filters are said to be finite because they do not have any feedback. Therefore, if you send an impulse through the system (a single spike) then the output will invariably become zero as soon as the impulse runs through the filter. A finite impulse response (FIR) filter is a filter structure that can be used to implement almost any sort of frequency response digitally. An FIR filter is usually implemented by using a series of delays, multipliers, and adders to create the filter's output. The difference equation that defines the output of an FIR filter in terms of its input is

$$y[n] = b_0x[n] + b_1x[n-1] + \ldots. + b_Nx[n-N]$$

Where:
x[n] is the input signal,
y[n] is the output signal,
b_i are the filter coefficients, and
N is the filter order – an Nth-order filter has (N + 1) terms on the right-hand side; these are commonly referred to as taps.

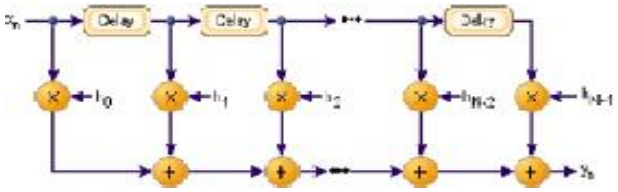


Fig. 2. Basic FIR filter strcuture

3. FILTER COEFFICIENTS

Filter coefficients can be derived from the impulse response or transfer function. The filter coefficients can be generated using many algorithms and the simplest algorithm that can be used is direct convolution FIR filter. The filter coefficients when represented in time domain, they characterize the system. The discrete time systems representation will be generally in the sampled form and coefficients represent the sample for finite impulse response filter.

3.1 Extraction of Filter Coefficients

The filter coefficients can be extracted by using the process called Remez Exchange Algorithm which is an optimal process. Function used to design the FIR filter coefficients using Remez Exchange Algorithm is called as firpm(). This firpm stands for Parks-McClellan optimal FIR filter design. firpm designs a linear-phase FIR filter using the Parks-McClellan algorithm. The Parks-McClellan algorithm uses the Remez exchange algorithm and Chebyshev approximation theory to design filters with an optimal fit between the desired and actual frequency responses. The filters are optimal in the sense that the maximum error between the desired frequency response and the actual frequency response is minimized. Filters designed this way exhibit an equiripple behavior in their frequency responses and are sometimes called equiripple filters. firpm exhibits discontinuities at the head and tail of its impulse response due to this equiripple nature. The syntax for the FIR filter is given as

2010-2014 – IJVES

Vol 06, Article 09634; October 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

follows.

. **b = firpm(n,f,a) (1)** returns row vector b containing the n+1 coefficients of the order n FIR filter whose frequency-amplitude characteristics match those given by vectors f and a.

The output filter coefficients (taps) in b obey the symmetry relation:

b(k) = b(n+2-k), k = 1,2,….,n+1

Vectors f and a specify the frequency-magnitude characteristics of the filter:

- f is a vector of pairs of normalized frequency points, specified in the range between 0 and 1, where 1 corresponds to the Nyquist frequency. The frequencies must be in increasing order.
- a is a vector containing the desired amplitudes at the points specified in f. The desired amplitude at frequencies between pairs of points ($f(k)$, $f(k+1)$) for k odd is the line segment connecting the points ($f(k)$, $a(k)$) and ($f(k+1)$, $a(k+1)$). The desired amplitude at frequencies between pairs of points ($f(k)$, $f(k+1)$) for k even is unspecified. The areas between such points are transition or "don't care" regions.
- f and a must be the same length. The length must be an even number.

3.2 Rounding of Coefficients in Matlab

The filter coefficients that will be obtained will be in floating point. For simple operations to be performed in Verilog with floating point values, they have to be converted into fixed point. The conversion from the floating point to fixed point will be done in Matlab by a function called round [2]. The syntax to be used is given as follows i.e. Y = round(X). The filter coefficients and the sine wave samples will be present in floating point up to four decimal places. So when the function round is used then it simply changes the coefficients into fixed point up to only one integer value. This one integer value is not sufficient to proceed to the next block of the multipliers and adders. So the values have to be increased to a larger value by multiplying the integers with 2 to the power of some other value. By doing so the integer values will be increased and so can be used in the next stages of the blocks.

3.3 Convolution in Matlab

The general FIR filter deals with the convolution part. In mathematics and, in particular, functional analysis, **convolution** is a mathematical operation on two functions f and g, producing a third function that is typically viewed as a modified version of one of the original functions, giving the area overlap between the two functions as a function of the amount that one of the original functions is translated. Convolution is similar to cross-correlation. It has applications that include probability, statistics, computer vision, image and signal processing, electrical engineering, and differential equations. The convolution of f and g is written $f*g$, using an asterisk or star. It is defined as the integral of the product of the two functions after one is reversed and shifted. As such, it is a particular kind of integral transform:

$$f(t) * g(t) = \int f(t-T)\, g(T)\, dT \qquad (2)$$

The convolution in Matlab is done by using a function called conv(). The general syntax in Matlab to be written is shown below.

$$w = conv(u,v) \qquad (3)$$

Specifications of FIR filter for design using Remez Exchange:

Passband edge frequency = 1500Hz
Stopband edge frequency = 2000Hz
Sampling frequency = 48000Hz
FIR filter tap = 16 Tap
Above passband edge= f1 = 3000Hz
Below passband edge = f2 = 300Hz

TABLE 1
CONVOLVED MATLAB VALUES FOR300 Hz

0	488	1566	2874	3912	4557	4771	4504
36	587	1736	3014	4019	4615	4755	4436
69	694	1916	3171	4101	4657	4744	4372
120	822	2071	3304	4200	4695	4717	4284
171	949	2235	3432	4284	4717	4695	4200
230	1084	2407	3557	4372	4744	4657	4101
310	1225	2566	3687	4436	4755	4615	4019
389	1384	2722	3801	4504	4771	4557	3912

2010-2014 – IJVES

Vol 06, Article 09634; October 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

The above table show some set of convolved output values i.e. when convolution is done between the FIR filter coefficients and input sine wave samples for a frequency of 300 Hz.

TABLE 2
CONVOLVED MATLAB VALUES FOR3000 Hz

0	288	612	915	1176	1353	1421	1368
1211	968	668	365	104	-73	-165	-157
157	453	685	811	811	685	453	157
-157	-453	-685	-811	-811	-685	-453	-157
157	453	685	811	811	685	453	157

The above table show some set of convolved output values i.e. when convolution is done between the FIR filter coefficients and input sine wave samples for a frequency of 3000 Hz.

4. IMPLEMENTATION

The task of implementing Verilog code is considered as the important one in the design of the FIR filter. The FIR Verilog code has to be written based on the FIR filter structure which is shown in the above fig. 2. As seen in the structure it consist of the multipliers and adders, the same has to be implemented in Verilog. The multipliers and the adders are to be replaced by their respective functions i.e. types of multipliers and adders that are used in this paper. First the coefficients have to be extracted from the Matlab which has been discussed above. Also the sine wave samples (extracted using Matlab) have to be considered as the inputs to the FIR filter. The coefficients will be extracted from Matlab with the specifications that are mentioned above. The Verilog code has to be written for all the combination of the multipliers and adders. Means each multiplier should be used as a combination with the other two adders. First the Verilog code will be written by considering the Remez Exchange Coefficients as the input coefficients to the FIR filter. Next the sine wave samples that are designed for two different frequencies (one above the passband frequency of Remez Exchange and the other below the passband frequency of Remez Exchange) are to be considered as inputs to the FIR filter. For this to be done the test bench in Verilog has to be written so that the final output will be obtained. This final output i.e. the values that are obtained have to be matched with the values obtained between the convolution of the Remez Exchange Coefficients and the Sine wave samples.

5. MULTIPLIERS AND ADDERS USED

5.1 Booth Multiplier

Booth's algorithm [4], [5] examines adjacent pairs of bits of the N-bit multiplier Y in signed two's complement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for i running from 0 to N-1, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator P remains unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product. The representation of the multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at $i = 0$; the multiplication by 2^i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest N bits of P. There are many variations and optimizations on these details. The algorithm is often described as converting strings of 1's in the multiplier to a high-order +1 and a low-order –1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

5.2 Wallace Tree Multiplier

A Wallace tree [6], [7] is an efficient hardware implementation of a digital circuit that multiplies two integers, devised by an Australian Computer Scientist Chris Wallace in 1964.
The Wallace tree has three steps:

2010-2014 – IJVES

Vol 06, Article 09634; October 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

1. Multiply (that is - AND) each bit of one of the arguments, by each bit of the other, yielding n^2 results. Depending on position of the multiplied bits, the wires carry different weights, for example wire of bit carrying result of $a_2b3$is 32.
2. Reduce the number of partial products to two by layers of full and half adders.
3. Group the wires in two numbers, and add them with a conventional adder.

The figure below shows how a Wallace Tree Multiplier can be realized for the 8-bit i.e. an 8x8 multiplier.

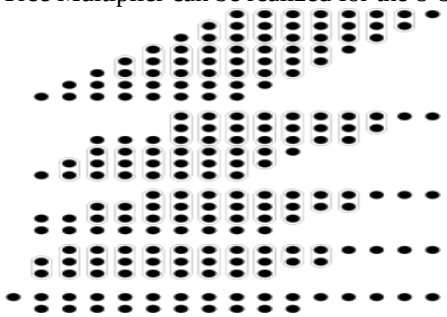


Fig. 3. Example of reduction on 8x8 multiplier

5.3 Carry Save Adder

A carry-save adder[7] is a kind of adder with low propagation delay (critical path), but instead of adding two input numbers to a single sum output, it adds three input numbers to an output pair of numbers. When its two outputs are then summed by a traditional carry-look ahead or ripple-carry adder, we get the sum of all three inputs. When adding three or more numbers together, a sequence of carry-save adders terminated by a single carry-look ahead adder provides much better propagation delays than a sequence of carry-look ahead adders. In particular, the propagation delay of a carry-save adder is not affected by the width of the vectors being added.

Implementation of Carry Save Adder

The carry-save [8] unit consists of *n* full adders, each of which computes a single sum and carry bit based solely on the corresponding bits of the three input numbers. Given the three *n* - bit numbers **a**, **b**, and **c**, it produces a partial sum **ps** and a shift-carry **sc**:

$$ps_i = a_i \oplus b_i \oplus c_i \qquad (4)$$

$$sc_i = (a_i \wedge b_i) \vee (b_i \wedge c_i) \vee (c_i \wedge a_i) \qquad (5)$$

The entire sum can then be computed by:

1. Shifting the carry sequence **sc** left by one place.
2. Appending a 0 to the front (most significant bit) of the partial sum sequence **ps**.
3. Using a ripple carry adder to add these two together and produce the resulting *n* + 1-bit value.

When adding together three or more numbers, using a carry-save adder followed by a ripple carry adder is faster than using two ripple carry adders. This is because a ripple carry adder cannot compute a sum bit without waiting for the previous carry bit to be produced, and thus has a delay equal to that of *n* full adders. A carry-save adder, however, produces all of its output values in parallel, and thus has the same delay as a single full-adder. Thus the total computation time (in units of full-adder delay time) for a carry-save adder plus a ripple carry adder is *n* + 1, whereas for two ripple carry adders it would be 2*n*.

5.4 Carry Skip Adder

The other name for the carry skip adder is the carry by-pass adder. The carry skip adders [9] take the advantage that the carry signal will be generated or else propagated. There are some special blocks where the circuit detects quickly if all the bits to be added are different. The signal will be produced by this circuit and known as propagation signal. If the carry is transmitted through all the stages in the block then the carry signal entering the block can directly be by-passed. Depending on the position at which a carry signal has been generated the propagation time will be varied. If suppose there is no carry generation then only the addition time will be considered which will be considered as the best case. Here in this project the design of 16 bit carry skip adder [13] is considered and hence the block diagram consists of four multiplexers. The diagram for the implementation of the Carry Skip Adder is shown below.

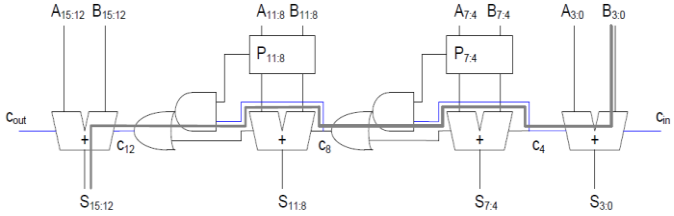


Fig. 4. Structure of Carry Skip Adder

2010-2014 – IJVES

Vol 06, Article 09634; October 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

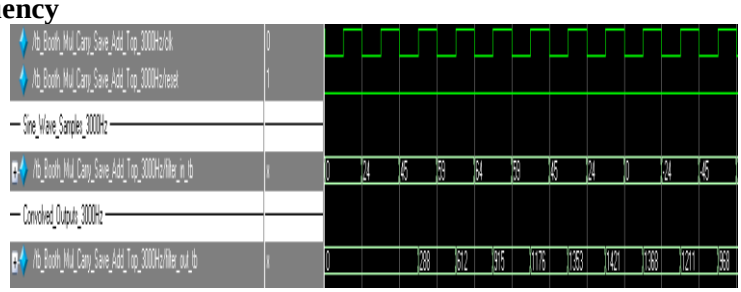
Implementation of Carry Skip Adder

The carry-out of each block is determined by selecting the carry-in and Gi:j using Pi:j. When Pi:j = 1, the carry-in cj is allowed to get through the block immediately. Otherwise, the carry-out is determined by Gi:j. The CSKA has less delay in the carry-chain with only a little additional extra logic. Further improvement can be achieved generally by making the central block sizes larger and the two-end block sizes smaller. Assuming the n-bit adder is divided evenly to k r-bit blocks, part of the critical path is from the LSB input through the MSB output of the final RCA. The first delay is from the LSB input to carry-out, which is 4r + 5. Then, there are k - 2 skip logic blocks with a delay of 3/\. Each skip logic block includes one 4-input AND gate for getting Pi+3:i and one AND/OR logic. The final RCA has a delay from input to sum at MSB, which is 4r+6.

6 SIMLATION RESULTS

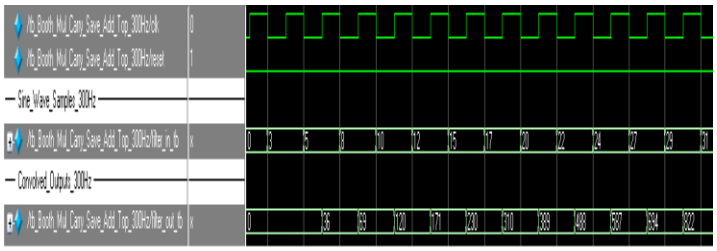
The simulation results will be obtained from Modelsim where in the output values will be matched with the convolved output values that have described in above tables 1 and 2. The simulation results will be done for both the frequencies i.e. 300 Hz and 3000 Hz. The simulation will be done for each combination of multiplier with adders that have been mentioned in this paper.

6.1 3000 Hz Frequency



The above waveform is for the combination of Booth multiplier and Carry Save Adder where the inputs (filter_in_tb) correspond to the sine wave samples for a frequency of 3000Hz that are present in the table 2. Similarly all the other waveforms will be obtained where the values will be matched.

6.2 300 Hz Frequency



The above waveform is for the combination of Booth multiplier and Carry Save Adder where the inputs (filter_in_tb) correspond to the sine wave samples for a frequency of 300Hz that are present in the table 1. Similarly all the other waveforms will be obtained where the values will be matched.

The above waveforms obtained from Modelsim have been shown for the case of Booth Multiplier with Carry Save Adder. This case has been shown for the frequencies of 300Hz and 3000Hz. Similarly the other three cases i.e. Wallace Multiplier with Carry Skip Adder, Wallace Multiplier with Carry Save Adder and the Booth Multiplier with Carry Skip adder (for 300Hz and 3000Hz) follows the same results.

6.3 Comparison between all multipliers and adders

The comparison between all the Multipliers and adders for this FIR filter can be done by taking the reports from the Device Utilization Summary and also the timing reports that will be obtained from the Xilinx ISE tool. The below table shows the comparison between all the multipliers and adders.

TABLE 3
COMPARISON BETWEEN MULTIPLIERS AND ADDERS

	DELAY TIME	MEMORY USED
FIR_BOOTH_MUL_CARRY_SAVE	71.545ns	259.27MB
FIR_BOOTH_MUL_CARRY_SKIP	56.003ns	263.77MB
FIR_WALLACE_MUL_CARRY_SAVE	51.070ns	216.27MB
FIR_WALLACE_MUL_CARRY_SKIP	34.499ns	251.96MB

2010-2014 – IJVES
Indexing in Process - EMBASE, EmCARE, Electronics & Communication Abstracts, SCIRUS, SPARC, GOOGLE Database, EBSCO, NewJour, Worldcat, DOAJ, and other major databases etc.,

Vol 06, Article 09634; October 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

7 CONCLUSION AND FUTURE SCOPE

The conclusion gives the information about the comparison for all the combination of multipliers and adders. The comparison is done by taking memory size and delay time into consideration.

7.1 Conclusion

From the above design of the FIR filter it can be concluded that the Wallace Tree Multiplier and the Booth Multiplier can be used. Also the two adders that can be used are carry save adder and carry skip adder. When these two adders are used with the combination of multipliers then it is been seen that for Wallace multiplier with carry save adder is occupying less memory when compared with carry skip adder and for Booth multiplier with carry save adder is occupying more memory when compared with carry skip adder. But the combination of multipliers with carry save adder is having more delay time when compared with the carry skip adder multiplier combination.

Among the multipliers used i.e. Wallace Tree Multiplier and Booth Multiplier, Wallace Tree Multiplier is used especially for small bit applications because as the bit width increases the design of Wallace Tree Multiplier will become more complex. At the same time Booth multiplier can be used for larger bit applications without any design complexity, by considering tradeoff factor for area when compared with the Wallace Tree Multiplier. So depending on the complexity and the area one can chose the corresponding multipliers for the design of FIR filter. For Wallace Tree Multiplier and Booth Multiplier with combination of carry save adder it is been seen that the memory usage for Wallace Multiplier has got 17% less memory when compared with Booth Multiplier. Similarly for the combination of multipliers with carry skip adder Wallace Multiplier has got 5% less when compared with Booth Multiplier.

Previous designs of the FIR filter include the array multiplier. But this array multiplier uses the dot method of multiplication operation with full adders and half adders. This has more complexity which is considered as a drawback in the FIR filter design. To overcome this drawback the Wallace Tree Multiplier has been invented (by Chris Wallace) where the design uses the full adders and half adders method by reducing the rows using 3:2 compressors. It is also seen that the FIR filter design with the array multiplier has got more amount of delay time when compared with Wallace Tree Multiplier. It is found that the FIR filter with Array Multiplier [10] has got 41.13ns. But for Wallace Tree Multiplier it is seen that there is less amount of delay time by 16%.

7.2 Future Scope

Booth multiplier uses the fixed width bit multiplication. Means there will be unnecessary computation in the partial products. This unnecessary computation cannot be truncated which will be considered as wastage of memory. Here there is no concept of configurability. This can be extended to the concept of Configurable Booth Multiplier which uses the exact number of bits by utilizing the optimum hardware. A design can be thought in such a way such that a configuration bit is available with the multiplier as well as multiplicand. The number of cycles in the multiplication can be reduced by introducing a technique called Dynamic Range Detection which will be used to truncate the unnecessary computation. This can be extended further by using Configurable Booth Multiplier with the combination of Wallace Tree Multiplier which gives accurate output with less amount of delay time with reduced area.

ACKNOWLEDGEMENTS

I wish to thank profusely and acknowledge with deep sense of gratitude to **M. Shyam Sunder (Assistant Professor)**, Department of Electronics and Communication Engineering, Vasavi College of Engineering for being the technical advisor during the course of project work for his valuable suggestions and encouraging me during the project work. His constant monitoring has enabled me to complete the project work successfully. I take this opportunity to acknowledge with thanks and deep sense of gratitude to **Dr. P.A.Govindacharyulu (Professor)**, Department of Electronics and Communication Engineering for giving valuable suggestions and encouraging me to complete the project work successfully. I express my gratitude and high regards to **Dr. K. Jaya Shankar**, **Head of the Department ECE**, and Vasavi College of Engineering for permitting me to carry out the project in the college campus. I would like to thank, **Dr. N.S.S. Reddy (Associate Professor)**, for his valuable support during my project work. Finally I would like to thank my parents, friends, classmates, all faculty members and non-teaching staff of ECE Department, Vasavi College of Engineering, and Hyderabad.

REFERENCES

[1] Animesh Panda, Satish Kumar Baghmar, Shailesh Kumar Agrawal, T.Siva Kumar, T. Usha "FIR Filter Implementation on A FPGA Allowing Signed and Fraction Coefficients with Coefficients Obtained Using Remez Exchange Algorithm", International Journal of Advancements in Technology

[2] Soo-Chang Pei, Chien-Cheng Tseng, "Two Novel Methods for Complex Allpass Filter Design Using Remez Exchange Algorithm", IEEE Transactions, Vol. 96, Page No. 3073-3078.

[3] Rabiner, L.R., J.H. McClellan, and T.W. Parks. "FIR Digital Filter Design Techniques Using Weighted Chebyshev Approximations." *Proc. IEEE* 63 (1975).

[4] Yi-Ting Chen, Wen-Shen Cheng, Jun Xian Teng, Shyh Jye Jou, "Sub Word and Reduced Width Booth Multipliers for DSP Applications", IEEE International Symposium, Vol. 3, III-575 – III-578.

2010-2014 – IJVES

Vol 06, Article 09634; October 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

[5] San Kim, Yong-Surk Lee, "A low Power Booth Multiplier using Novel Data Partition Method", IEEE Advanced System Integrated Circuits (2004), Page No. 54-57
[6] Waters, Ron S, "A Reduced Complexity Wallace Multiplier Reduction", IEEE Transactions on Computers, Vol. 59, Page No. 1134-1137
[7] C. S. Wallace,"A Suggestion for a Fast Multiplier," IEEE Trans. Electronic Computers, vol. 13, pp. 14 – 17, 1964.
[8] T.G.Noll, "Carry-Save Architectures for High Speed Digital Signal Processing", IEEE Transactions on VLSI Signal Processing, Vol.3, pp.121-140, 1991.
[9] A.Guyot, B.Hochet, and J-M Muller, "A Way to Build Efficient Carry Skip Adders", IEEE Transactions on Computers, October 1987.
[10] S.Karunakaran, N.Kasthuri "FIR Filter Design using Array Multiplier", European Journal of Scientific Research, Vo.67, pp.625-635, 2012.

2010-2014 – IJVES