International Refereed Journal of Engineering and Science (IRJES)
ISSN (Online) 2319-183X, (Print) 2319-1821 *Volume 1, Issue 1 (September 2012), PP.031-038*
www.irjes.com

Design of FIR Filter by using Sharing Multiplier with Low Delay

*Kantamaneni. Sravanthi, **Dr.V.V.K.D.V.Prasad Battula, ***Veera Vasantha Rao
*Lecturer ,E.C.E Dept
**Professor,E.C.E Dept.,
***Lecturer,E.E.E. Dept.,.

Abstract : *(Keywords: CSHM, WTM) Finite Impulse Response (FIR) filtering operation can be expressed as Multiplication of vectors by scalars. A high speed design for FIR filters based on a Computation Sharing Multiplier (CSHM) is presented here. This CSHM [1,5], which specially targets computation re-use in vector-scalar products. Our recently proposed Computation Sharing Multiplication approach can be effectively used to reduce redundant computations in FIR filtering operation.*

The main idea is to represent the multiplication in FIR filtering operation as a combination of add and shift operations over the common computation results. The common computations are identified by decomposing the coefficients of FIR filters. These computations are performed only once and shared Multiplier approach that achieves high performance in FIR filtering with less overhead. The performance of the proposed implementation is compared with implementations based on multipliers like Wallace Tree Multiplier (WTM) (Delay 13.2ns & Area 146 LUTS) & Booth Multiplier(Delay 26.642ns & Area 180 LUTS) .We show that sharing multiplier scheme improves the parameters like Delay & Area with respect to the FIR filter implementations based on the Booth Multiplier & Wallace tree Multiplier.

I. Introduction

Recent advances due to the popularity of the portable battery-powered wireless communication systems such as cellular phones, pagers and wireless modems and multimedia applications demand high performance and low-power VLSI Digital Signal Processing (DSP) is Finite-Impulse Response (FIR) filtering. The FIR filter performs the weighted summations of input sequences, which are frequently used to implement high pass, low pass and many other types of filters and is widely used in video convolution functions, signal preconditioning and various communication applications.

Recently, due to the high-performance requirement and increasing complexity of DSP and multimedia communication applications, FIR filters with large filter taps are required to operate with high sampling rate, which makes the filtering operation very computationally intensive. Complexity reduction of FIR filter implementations has also been of particular interest since lower computational complexity leads to high performance as well as low delay & area design.

The ongoing FIR filter project is a case design of a standard signal processing block employing high performance VLSI design techniques and methodologies. The Current prototype has been designed using techniques found in high performance applications including hard disk controllers, DSP processors and data acquisition systems. The most important and expensive operation performed by the FIR filter is the Digital Signal Processing (DSP) function. Besides the MAC circuitry, the FIR filter is a fairly simple system both in concept and implementation.

Furthermore, the only other circuitry that are required are registers to hold the coefficients, simple control logic and pipeline latches if pipelining is to be realized. Therefore, in the design of an FIR filter most of the design effort should and has been focused on optimizing the MAC circuit. For a filter input size of N>4 the multiplier becomes extremely expensive in respects to delay and power consumption and defines the worst-case operation speed. The prototype is used solely for my observation of results in using different subsystem designs. Several techniques have been proposed in literature to achieve high performance and low power implementation of FIR filters with discrete coefficient values selected from the powers-of-two coefficient space .canonical-sign-digit and distributed arithmetic are widely used in the FIR filter design with fixed coefficients. Using those techniques, the FIR filtering operation can be simplified to add and shift operations. One approach uses integer linear programming to search for the optimized discrete filter coefficient that confirms to desired frequency response.

The major difficulty encountered in this scheme is that as the filter size or the number of bits used to represent a coefficient significantly increase, a large amount of computation is needed. Computation reduction techniques in digital FIR filters, which reduce redundant computation, have also been proposed. This approach computes the filter output using coefficient differences instead of their original values and decreases the redundant computation by computation reordering and sharing. However, additional computational overhead is introduced for identifying the common computations, the results of which can be shared by the sequence of operations. This approach also needs additional memory area for computation sharing. To achieve high performance without additional memory area computation sharing multiplier is used. Delay, area of the proposed implementation is compared with those of FIR filter implementations based on Booth Multiplier and Wallace Tree Multipliers.

In this paper, a simple scheme in which common computations can be identified with less overhead and shared without additional memory area is explored. A computation sharing multiplication approach, which can be effectively used to reduce redundant computations in FIR filtering operation, is presented here.

The main idea is to represent the multiplication in FIR filtering operation as a combination of add and shift operations over the common computation results. The common computations are identified by decomposing the coefficients of FIR filters. These computations are performed only once and shared in parallel without using memory. This sharing property enables the computation sharing multiplier approach that achieves high performance in FIR filtering with less overhead.

II. FIR Filtering Operation Introduction

The basic operation of an FIR filter is to compute weighted sums of an input signal by performing a series of Multiply And Accumulate (MAC) operations on the input signal. DSP operations most widely use FIR filtering. Each MAC operation is performed in exactly one stage of the FIR filter. Within a stage the input signal is multiplied by a constant coefficient. The product of the input signal and the coefficient is then added to the accumulated multiply and accumulate value of the previous stages and the previous inputs. The behavior of an output y(n) of an FIR filter with N stages and inputs x(n) and coefficients c(k) can be shown as:

$$Y(n)=c(0)x(n)+c(1)x(n-1)+c(2)x(n-2)+\ldots\ldots\ldots\ldots+c(N-1)x(n-N)$$

In hardware each stage of an FIR filter is computed with in a TAP. A TAP consists of a loadable coefficient register, two input registers, a multiplier, an adder and an output register. In FIR filtering, depending up on the number of taps the delay is calculated. The FIR filter has N taps, the delay is :(N-1)/(2*Fs); for eg:- A 21 tap FIR filter operating at a 1KHZ rate has delay of 10ms.For an 8-tap, 8-bit coefficient FIR filter, eight of the above modules are cascaded to create the design. The coefficient register is 8 bits wide to store the 2's complement signed coefficient value of each stage. Each TAP consists of multiplier, adder, coefficient register, input registers and sum registers. Once the basic block has been created in maximum and verified in HSPICE the FIR filter will be created by arraying and interconnecting eight TAP modules. Dividing the FIR filter into an array of TAPS as apposed to creating one big FIR block allows modularity and flexibility. The modularity with in the TAP sub block allows exchanging of sub blocks (i.e. adder, multiplier) with varying configurations to obtain the best area/speed/power design. This TAP module was then instantiated eight times to create the FIR_FILTER for functional verification.

III. Vector Scaling Operation

FIR filtering operation can be expressed as multiplications of vectors by scalars. The input-output relationship of Linear Time Invariant (LTI) FIR filter can be described as

$$Y(n) = \sum_{K=0}^{M-1} C_K X(n-k)$$

Where M represents the length of FIR filter, C_K'S are the filter coefficients, and x (n-k) denotes the data sample at time instance (n-k).

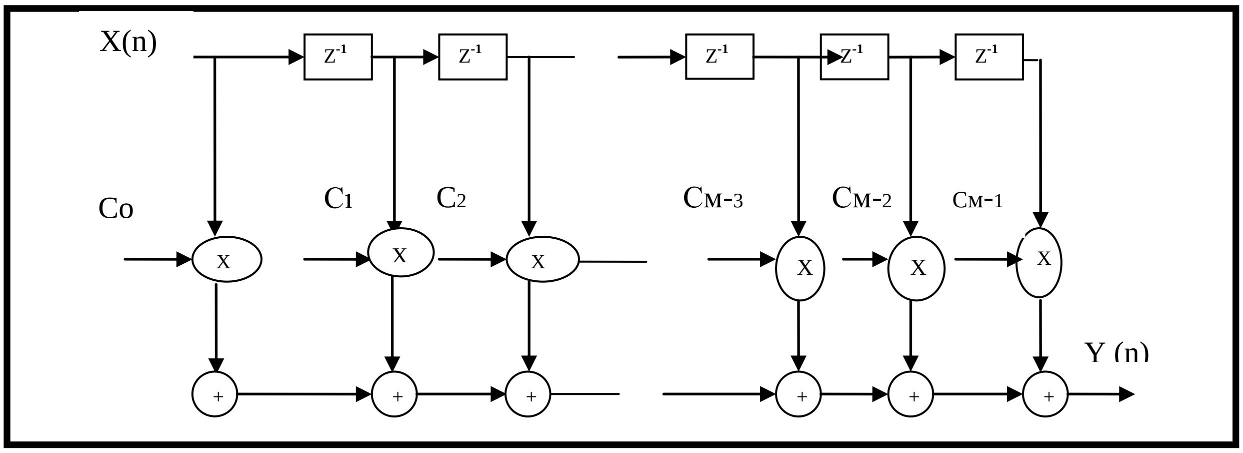
FIR Filter Methods:

Two common methods of realizing FIR filters are the Direct Form (DF) and the Transposed Direct Form (TDF) .In the DF realization of the filter there are delay units between multipliers. This implies that the present input ,x(n) and N-1 previous samples of the input, that is x(n-1) to x[n-(N-1)],are applied to each multiplier inputs and the outputs of these multipliers are summed together to form the filter output y(n).

$$Y = X\,.C$$

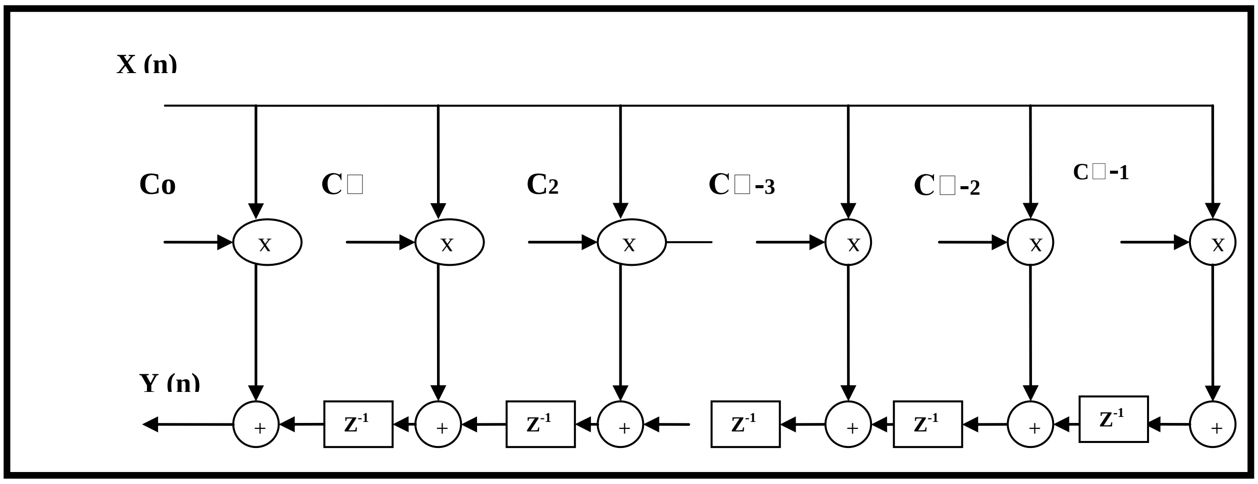
In the TDF, however, delay units are placed between adders so that the multipliers can be fed simultaneously. A DF filter is implemented such that at each clock cycle a new data sample and the corresponding filter coefficient are applied to the multipliers inputs. This Continuous change at both inputs will cause a high level of switching activity within the multiplier, and hence leading to higher power consumption.

Fig 1.1: Direct Form FIR Filter



Using a TDF realization, however, the switching activity at data inputs of the multiplier is significantly reduced since the data input remains unchanged for a significant number of multiplication operations. This results in a considerable reduction in switching activity within the multiplier circuit and consequently leads to less power consumption than the DF realization.
A further reduction in switching activity within the multiplier section of both DF and TDF filters can be achieved by implementing the filters such that respective
A further reduction in switching activity with in the multiplier section of both DF and TDF filters can be achieved by implementing the filters such that respective data samples are multiplied with filter coefficients in a non-sequential order. This results in reducing the switching activity at both multiplier inputs. In the Direct Form FIR filter, a large adder in the final stage lies on the critical path and it slows down the FIR filter. The Transposed Direct Form FIR filter is more appropriate for a high-performance filter structure.Figure1.1 shows a direct form implementation of an FIR filter. An equivalent architecture is the transposed direct form as shown in figure1. 2. The TDF implements a product of the coefficient vector C = [Co, C1……. C M-1] with the scalar x (n) at time n. The input x (n) is multiplied by all the coefficients Co, C1……. CM-1 simultaneously. In the sequel, such product will be referred to as a vector scaling operation. Expressing the filtering operation in terms of a vector scaling operation allows opportunity to share computations between operations. The method to identify common computations, which can be shared amongst a sequence of operations in the vector and scalar product, is also described.

Fig 1.2: Transposed Direct Form FIR Filter



As a result, the multiplication operation is significantly simplified as add and shift operations. Complexity reduction in the vector scalar product can be achieved by using the concept of computation sharing. In the direct form FIR filter, a large adder in the final stage lies on the critical path and it slows down the FIR filter. Since we focus on the design of a high performance FIR filter, the transposed direct form FIR filter is more appropriate for a high-performance filter structure.

IV. Computation Sharing Multiplier Implementation

The Computation Sharing Multiplier (CSHM)[1,5] architecture and implementation, which is based on the algorithm is presented. CSHM consists of Precomputer Select Units And Adders (S&A). The precomputer produces the multiplication of alphabets with input x and the S&A performs the add and shift operations required to obtain the final output.

The main advantage of CSHM is that the outputs of precomputer are shared by all the select units. Hence, the operations performed by select units can be executed in parallel without introducing additional select unit delay.

The Booth Multiplier and Wallace Tree Multiplier (WTM) is also implemented to compare these multipliers in terms of delay and area. In this design an 8x8 Computation Sharing Multiplier structure is been implemented. Depending on the length of the coefficient (W) select units will be selected. This CSHM algorithm is applicable to only unsigned multipliers. In this Computation Sharing Multiplier design, the fixed-size look-up rule is used for different select and shift signal for different coefficient inputs. In the fixed size look-up multiplication, the maximum alphabet length L is fixed.

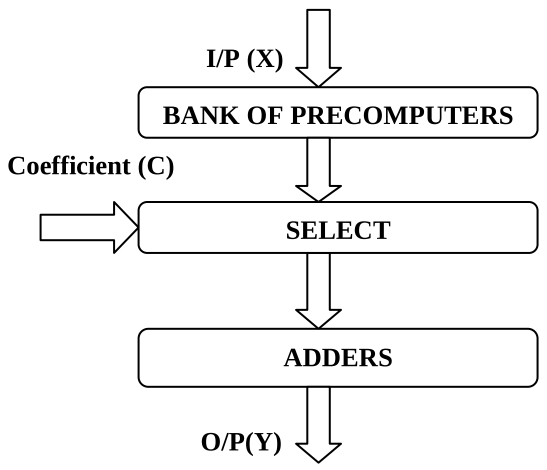


Fig:2.2 : CSHM Structure

The coefficient input, the length of which is W, is divided in to W/L parts and each part consists of L bits. This implies that W/L should be an integer. Once L is determined, an alphabet set should be able to express any L bit number by one of its alphabet multiplied by the powers of 2 (shift operation). The add operations on these parts to generate the final result follow the shift operations. One possible alphabet set consists of odd numbers that are less than or equal to 2^L-l. These numbers cover all the coefficients up to bits using only shift operations. With fixed value of L, these numbers from one of the alphabet sets with minimum cardinality. For instance, the alphabet sets obtained for L=2 and 4 are {1, 3} and {1, 3, 5, 7, 9, 11, 13, 15}, respectively. The advantage of the fixed size look-up rule is that much of the operation up to final additions can be performed in parallel. The computation sharing multiplier design with set to 4 provides adequate tradeoff between the number of addition and multiplication operations.

Hence, we use the alphabet set {1, 3, 5, 7, 9, 11, 13, and 15} in our multiplication.

V. CSHM Algorithm[1,5]

In vector scaling operations, select a set of small bit sequences so that the same multiplication result can be obtained by only add and shift operations. For instance, (1011).X can be decomposed as 2^0 (0011).X + 2^3 (0001).X. If both (0011) X and X are available, the entire multiplication process is reduced to a few add and shift operations.

The basic bit sequences are referred as alphabets. Also, an alphabet set is a set of alphabets that spans all the coefficients in vector C. The ith coefficient can be obtained in the same manner. It can be represented as

$$C_i = \sum_{K=0}^{L} 2^{m_k} \alpha_{k,j}$$

Where mĸ is a shift value and αk, j are alphabet that belong to the jth for k = 0, 1, 2 …L. alphabet set, for.

If we multiply the scalar X to both sides the multiplication C_i.**X** can be expressed as

$$\mathbf{C_i. X = \overset{L}{\underset{K=0}{\square}} \; 2^{m}k \, \square \, k, j. X}$$

Hence multiplication C_i. X can be significantly simplified to add and shift operations of αk, j. X, which is multiplications of and all the elements of the predetermined alphabets. Since alphabets are small bit sequences, the multiplication of alphabets with the operand X can be done with out seriously compromising the performance.

Coefficient Decomposition

Coefficient decomposition is done in order to identify the common computations. Depending on the selection of the alphabet set, the number of required add and shift operations changes. Obviously, an alphabet set should cover all the coefficients in coefficient vector C. As the number of coefficients in C increases, there can be many choices for alphabet sets on the coefficients and each alphabet set gives rise to a different combination of add and shift operations to obtain C.X. In addition, there are two other desirable characteristics of 'good' alphabet set. First, total numbers of add operations should be minimized. Multiplication operation can be simplified to add and shift operations with the computation sharing multiplier algorithm.

In the multiplier implementation, the add operations lie on the critical path and incurs the largest delay. Therefore, reduction of the number of adds operations improves overall performance. Second, the number of alphabets in alphabet set should be minimized. In the computation sharing Multiplier scheme, the multiplied value of αk, j. X should be available before the decomposition. They are computed at the first stage. As the number of alphabets increase, the amount of the computations also increases, which results in large area and power consumption. It also increases delay due to large length of individual alphabet. Generally, these two properties conflict with each other. A larger set of possible decompositions is possible as the number of alphabets increases. Therefore, if the alphabets are selected properly, large number of alphabets give rise to less number of add operations. Likewise, the reduction in number of alphabets results in many adds operations.

CSHM Architecture[1,5]

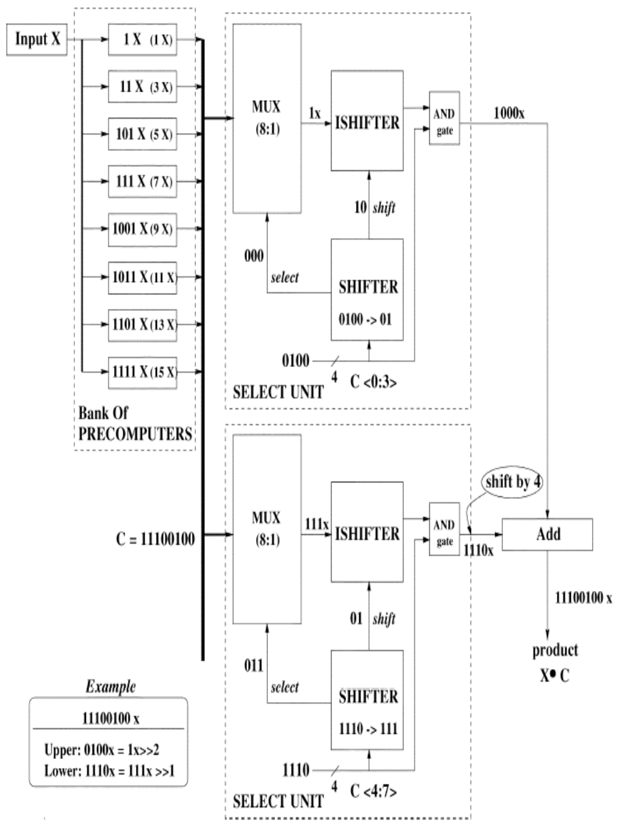


Fig 2.2:Parallel 8x8 CSHM Architecture[1,5]

In this section, a parallel 8X8 CSHM structure based on the scheme is described. The bank of precomputer performs the computations αk. X, K = 0, 1, 2……….8. As a result, the outputs of the precomputer bank are 1x, 3x, 5x, 7x, 9x, 11x, 13x and 15x. To find the correct alphabet, SHIFTERs perform the right shift operation until it encounters 1 and send an appropriate select signal to 8-to-1 (8:1) Muxes. It also sends the exact shifted values (shift signal) to ISHIFTER's. The 8-to-1 (8:1) Muxes select the correct alphabet among the eight values received from precomputer, **k** X, K = 0, 1, 2………8. The ISHIFTER's simply inverse the operation performed by shifters. When the coefficient input is 0000, a zero output with the shifted value of the precompiled outputs is obtained.

Select and Shift Signal for Different Coefficient Inputs:

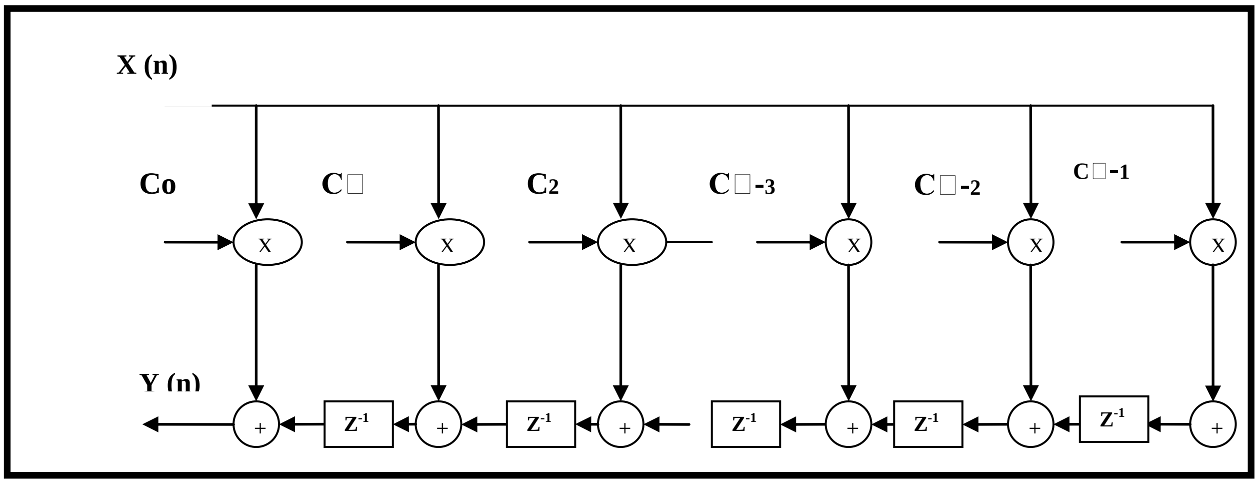
Shifter Input	Select Signal	Shift Signal
0000	Ddd	Dd
0001	000	00
0010	000	01
0011	001	00
0100	000	10
0101	010	00
0110	001	01
0111	011	00
1000	000	11
1001	100	00
1010	010	01
1011	101	00
1100	001	10
1101	110	00
1110	011	01
1111	111	00

Simple AND gates are used to deal with zero (0000) coefficient input. SHIFTER- MUX (8:1) – ISHIFTER – AND gate are referred as the select unit. The Upper Select Unit generates the multiplication of 4 LSBs of the coefficient with the input x. The lower select unit produces the product of upper 4 bits with input x. A shift of the upper 4 bits is performed when those two values are fed to the adder. A simple adder produces the final result. Consider an example as shown in figure 4.2. If the coefficient is 11100100, it is divided in to two parts consisting of 4 bits. 0100 is fed to SHIFTER of the upper select unit and 1110 to that of the Lower Select Unit. In the upper select unit, SHIFTER shifts 0100to the right twice until it encounters 1 and it sends 000 (select signal) to MUX (8:1), which chooses 1x among the precomputer outputs. SHIFTER also sends 10 (shift signal) to ISHIFTER. ISHIFTER shifts to the left input from the MUX (8:1) 1x twice. Finally, output of ISHIFTER is 0100. In the lower select unit, SHIFTER shifts 1110 to the right once and sends 011 (select signal) to MUX (8:1), which chooses 111 x among the outputs of precomputer. Like the one in the upper select unit, SHIFTER sends 01 (shift signal) to ISHIFTER, which shifts 111x to the left once. the AND gates of both select units just pass their inputs. The outputs of the upper select unit and the lower select unit are 0100x and 1110x, respectively. When these values reach the adder, 1110xshould be shifted four times to the left because it is the multiplication of the four MSBs. The precomputer, MUX (8:1), ISHIFTER AND gates, and ADDER lie on the critical path in this multiplier structure. The values for select and shift signal from SHIFTER for different coefficient inputs is shown in Fixed-size look up rule given in the table below. In the CSHM, when a coefficient is larger than 8 bits, additional select units will be added. The outputs of the precomputer are shared by all the select units and the operations performed by the select units are done in parallel.

VI. FIR Filter Implementation Using CSHM

FIR Filter Implementation

The computations αk.X, k = 0, 1, 2 …8, are performed only once for all k's and all filter taps and these values are shared by all the select units for generating C_i. X, i = 0, 1, 2 …8. In CSHM, precomputer delay is smaller than that of select units and adders. Only select unit and adder lie on the critical path. The SHIFTER delay is excluded from the critical path because the filter coefficient C_i is provided to the corresponding SHIFTER earlier than the input x. The proposed CSHM architecture has performance and power advantage

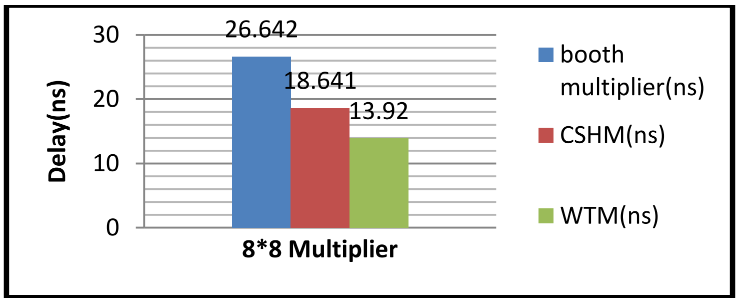


through the additional pipelining and the sharing of the precomputer outputs by all the select units and adders.

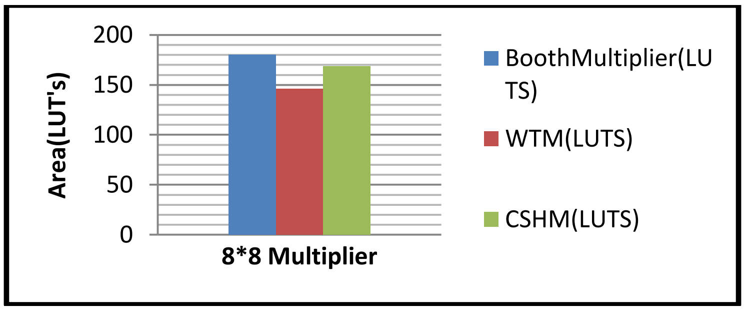
VII. Designing, Synthesis & Results

First, VHDL is used for circuit modeling. After checking logic function with Modelsim we synthesized and optimized the VHDL model. The results of delay, area of each multiplier is shown in the following graphical representations. Wallace tree multiplier has better performance than Computation Sharing Multiplier. The area and power of CSHM is larger than those of Booth Multiplier and WTM. As, a single multiplier, CSHM is not practical because of the large power consumption and area. The advantage of CSHM can be maximized when it is applied to the FIR filtering operations.

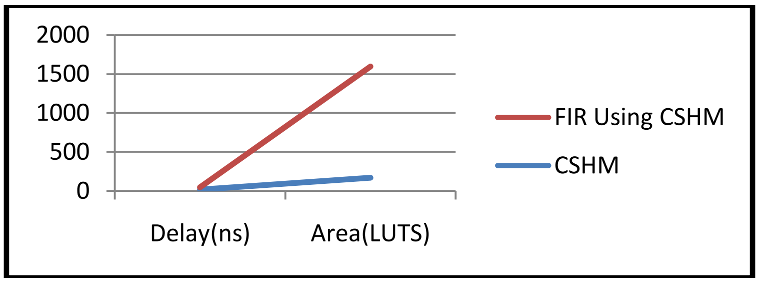
Delay of Each Multiplier



Area of Each Multiplier



Delay & Area of FIR Using CSHM



VIII. Conclusion

In this work the architecture of computation sharing multiplier for high performance FIR filter is presented. A multiplication approach, which specially targets the reduction of redundant computation in FIR filtering. By, using a CSHM algorithm, the multiplications in the vector scaling operation can be significantly simplified to add and shift operations of alphabets multiplied by input x. These common computations can be shared by the sequence of operations in vector scaling operations. The performance of the CSHM is better than commonly used Booth Multiplier(BM) . Because when compare the CSHM with the already existing systems of BM(Delay of **26..642ns & area is 180 LUTS)** and WTM (**13.92ns and146 LUTS**) the delay is lesser one of (**18.641ns**) and area occupied by this multiplier is of (**169 LUTS**).FIR filter implementation based on the proposed scheme shows higher performance and comparable Area with FIR filters based on BM and WTM. The CSHM scheme exhibits better power delay than other multiplier schemes. CSHM scheme leads to higher performance in filtering operation with out incurring large register overhead. The FIR structure based on CSHM can also be used in adaptive filter application. The idea behind in this work can assist design of DSP algorithms and their implementation for high-speed application.

VIIII. Future Scope

In this work the heart of the Finite Impulse Response Filter is the Computation Sharing Multiplier. In the multiplier the multiplexer used to be of size equal to 8:1, which is for the selection of outputs from the Precomputer block. If the coefficient length is more than 8 bit at that time the multiplexer size must be more than 8:1. That implementation going to deals with more delay, power consumption and area.

References

[1.] Lim Y.C. and S.R. Parker,” FIR filter design over a discrete power of two coefficient space,” IEEE Trans. Acustics, speech signal processing, vol. ASSP-31, pp.583 -591 , June 1983.

[2.] Mark Ronald Santoro, Design And Clocking of VISI Multipliers, Technical Report No. CSL-TR-89-397 october 1989.

[3.] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective* Englewood Cliffs, NJ: Prentice-Hall, 1996.

[4.] IEEE Transactions on circuits and systems, vol 44, No 6,June 1997.

[5.] Muhhammad k, “ Algorithmic and architectural techniyues for low power digital signal processing ,” ph.D. dissertation , purde univ., Hammond, IN, 1999.

[6.] T.Sheuan Chang, Yuan- Hua Chu ,”Low-Power FIR filter realization with differential coefficients and inputs,” IEEE Transactions on analog and digital signal processing, vol 47.No.2.February 2000

[7.] Multiplication in FPGA’s “ the performance FPGA DESIGN Specalist,” Andraka consulting group, Inc.

[8.] Park, J., H. choo, K.Muhammad, and K. Roy,” Non adaptive and Adaptive filter implementation based on sharing multiplication,” in proc . IEEE Int. conf. Acoustics, speech and signal processing (ICASSP), Istanbul, Turkey, june 2000.

[9.] Rabaey J.M; Digital Integrated circuits . A Design perpective Englewood cliffs, NJ: prentice-Hall, 1999 & Roy k. and s. Prasad, low power CMOS VLSI circuit Design, 1st ed. New york: Wiley, 2000.

[10.] Choo H., K.Muhammad, and K.Roy, “ Decision feed back equalizer with two’s complement computation sharing multiplication,” in proc. IEEE Int. conf. Acoustics, speech and signal processing [ICASSP], 2001.

[11.] J.Park, Woopyo Jeong, H.Choo, ”High performance and low power FIR filter design based on sharing multiplication,” in ISPLED August 2002.