International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 02, February-2015

High Speed and Multiplierless Implementation of Half-Band Filter

Divya Naga Padmini P
M.Tech (V.L.S.I) student
Department of ECE, GITAM University
Hyderabad, Telangana, India

***Abstract*—Half-band FIR filters utilize less hardware compared to normal FIR filters. The efficiency of half-band filters derives from the fact that about half of the filter coefficients are zero, thus, cutting down the implementation cost. Because of usage of multipliers in a FIR filter design gives rise to few demerits in terms of increase in area and increase in the delay which ultimately results in less performance. To resolve this issue, DAA (Distributed Arithmetic Architecture) is used which is a popular method for implementing digital FIR filters on FPGAs through which delay can be reduced and multiplierless realization can be achieved. The paper represents different ways of realizing half-band FIR low pass filter and provides comparison of critical path delay and clock frequencies for direct form, transposed form and DA (Distributed Arithmetic) type of architectures is mentioned by using of XILINX ISE 9.2i tool, for simulation and synthesis.**

***Keywords*—Half-band FIR filter, Direct form realization, Transposed form realization, DAA, critical path delay.**,

I. INTRODUCTION (*INTRODUCTION*)

The FIR halfband filter is simple to design, hardware cost effective and has moderate coefficient sensitivity. The cutoff frequency for a halfband filter is always π /2. Moreover, the passband and stopband ripples are identical, limiting the degrees of freedom in the design. Halfband filters have two important characteristics, the passband and stopband ripples must be the same, and the passband-edge and stopband-edge frequencies are equidistant from the halfband frequency π /2. Linear-phase FIR half-band filters have found several applications in the past. For instance, in the design of sharp cutoff FIR filters, a multistage design based on half-band filters is very efficient [3]. The efficiency of half-band filters derives from the fact that about 50 percent of the filter coefficients are zero, thus, cutting down the implementation cost [1], [3]. Half-band filters have also been used in multirate filter bank applications, either directly or indirectly. H (z) denotes the transfer function of a (linear-phase, FIR) half-band filter of order N – 1

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}, \qquad h(n) \text{ real.} \tag{1}$$

This paper presents an efficient implementation of Finite Impulse Response (FIR) half-band filter using Distributed Arithmetic (DA) architecture. Here, the multipliers in FIR filter are replaced with multiplierless DA based technique [5], [6]. The DA based technique consists of Look Up Table (LUT), shift registers and scaling accumulator. This architecture provides an efficient area-time-power implementation which involves significantly less latency and less area-delay complexity when compared with existing structures for FIR Filter like direct form and transposed form realizations.

II. HALF-BAND FILTER

Nyquist filters are a special class of filters which are useful for multirate implementations. Nyquist filters also find applications in digital communications systems where they are used for pulse shaping (often simultaneously performing multirate duties). Nyquist filters are also called Lth-band filters because the passband of their magnitude response occupies roughly 1/L of the Nyquist interval. The special case, L =2 is widely used and is referred to as Half band filters. Halfband filters can be very efficient for interpolation/decimation by a factor of 2. Moreover, the passband and stopband ripples are identical, limiting the degrees of freedom in the design. A half-band filter is a low pass filter that reduces the maximum bandwidth of sampled data by an approximate factor of 2 (one octave). When multiple octaves of reduction are needed, a cascade of half-band filters is common. And when the goal is down sampling, each half-band filter needs to compute only half as many output samples as input samples.

The frequency response of half-band filter is of the form

$$H(e^{jw}) = e^{-jw.(N-1)/2} Ho(e^{jw})$$

where Ho (e^{jw}) represents the real-valued amplitude response. A typical plot of Ho(e^{jw}) is shown in Fig. 1, assuming an equiripple type of design. There is symmetry with respect to the half-band frequency π /2, i.e., the band edges are related as

$$\omega_p + \omega_S = \pi \tag{2}$$

and the ripples are related as

$$\delta_1 = \delta_2 = \delta \tag{3}$$

where Wp and Ws represent passband ripple and stop band ripple of a half-band filter. The amplitude response of half-band filter Ho(e^{jw}) is shown in Fig.1.

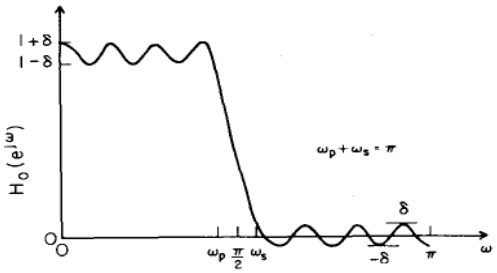

Fig.1. Amplitude response of a half-band FIR filter

In view of this symmetry, the impulse response h(n) satisfies

$$h(n) = \begin{cases} 0, & n - \dfrac{N-1}{2} = \text{even and nonzero} \\ \dfrac{1}{2}, & n = \dfrac{N-1}{2} \end{cases} \quad (4)$$

The Fig. 2 shows the half band filter response which clearly shows that half of the coefficients are reduced when compared with a normal FIR filter. The Half-band FIR filter has a main advantage that is half of the coefficients are reduced compared to that of a FIR filter which results in having low power consumption, higher operating speeds and smaller area.

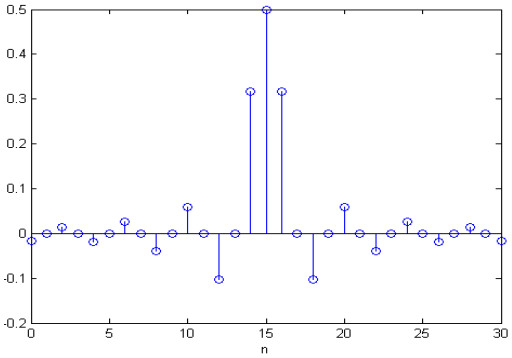

Fig. 2. Half band filter impulse response

III. HALF-BAND FILTER REALIZATION IN DIFFERENT FORMS

A. *FIR direct form realization-*

A set of inputs are shifted through number of registers also called taps and then multiplied by a number of constant filter coefficients (h[0], h[1]..,h[M]) and the sum of all these partial products results in y[n] which is filtered output. Basically a FIR filter is a data stream multiplied by a set of constants. The Fig. 3 describes the direct form of realization for FIR filter. Eq. 5 shows the direct form realization of FIR filter mathematically.

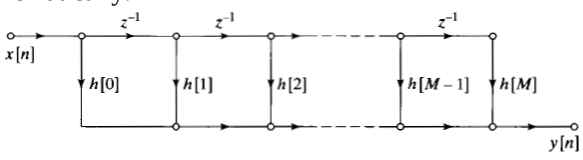

Fig. 3. Direct form realization of FIR filter.

$$y[n] = \sum_{k=0}^{M} h_k x[n-k] \quad (5)$$

B. *FIR transposed form realization-*

Transposed form is self pipelined with the cycle period the delay of an adder and a multiplier. But it has more area than directed form. The Fig. 4 shows the transpose form structure for the realization of FIR filter. The transposed forms result from the transposition theorem from signal-flow graph theory, which states that in a signal-flow graph if

- The arrows on all graph branches are reversed.
- Branch points become summers, and summers become branch points.
- The input and output are swapped and then the input/output relationships remain unchanged. The same applies to block diagrams.

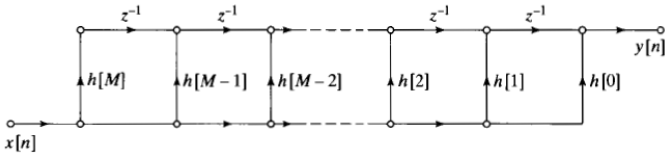

Fig. 4. Transposed form realization of FIR filter

There are many reasons why FIR filters are very attractive for digital filter design. Some of them are:

- Simple robust way of obtaining digital filters
- Inherently stable when implemented non recursively
- Free of limit cycles when implemented non recursively
- Easy to attain linear phase
- Simple extensions to multirate and adaptive filters
- Relatively straight-forward to obtain designs to match custom magnitude responses

C. *Distributed Arithmetic Architecture*

An efficient implementation of Finite Impulse Response Filter (FIR) can be attained using Distributed Arithmetic (DA) architecture. Here, the multipliers in FIR filter are replaced with multiplierless DA based technique [2], [5]. The DA based technique consists of Look up Table (LUT), shift registers and scaling accumulator. The architecture provides an efficient area-time-power implementation which involves significantly less latency and less area-delay complexity when compared with other structures of FIR Filter.

The Fig. 5 [5],[7] represents the normal realization of FIR filter that is multiplying input coefficients (x(n)) and filter coefficients (c(n)) and then summing them. In figure.2.6 representation involved in DA architecture implementation is described.

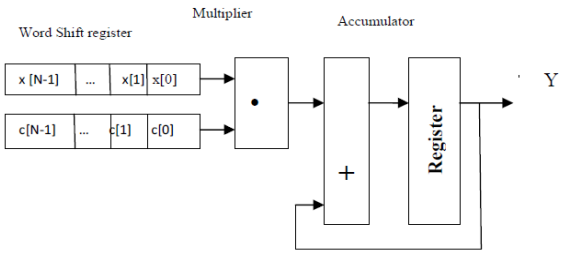

Fig. 5. Conventional implementation of FIR filter.

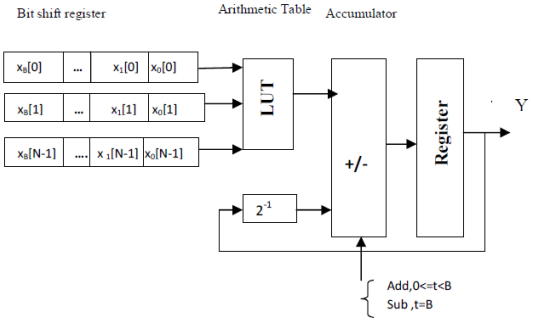

Fig. 6. Implementation of FIR filter using DA architecture.

Design Procedure for DAA realization-

Step1: Derive the filter coefficient according to specification of filter.

Step2: Store the inputs value in input register

Step3: Design the LUTs as shown in Table I, which represents all the possible sum combination of filter coefficients.

Step4: Accumulate and shift the value according to partial term beginning with LSB of the input and shift it to the right to add it to the next partial result.

Step5: First value must be subtracted, due to negative bit of MSB.

Step6: Analyze the output of filter as per specifications, otherwise go to step 1.

Step7: The same procedure is applied for Parallel DA FIR from step 1 to step 6 except in Step3 where bit address value is being called for 2-bits at single time, so that two times LUT is required in comparison to the serial DA.

TABLE I.
Look up table (LUT) for a 4tap filter with all possible combinations where h denotes filter coefficients.

ADDRESS	DATA
0000	0
0001	H_3
0010	H_2
0011	H_2+H_3
0100	H_1
0101	H_1+H_3
0110	H_1+H_2
0111	$H_1+H_2+H_3$
1000	H_0
1001	H_0+H_3
1010	H_0+H_2
1011	$H_0+H_2+H_3$
1100	H_0+H_1
1101	$H_0+H_1+H_3$
1110	$H_0+H_1+H_2$
1111	$H_0+H_1+H_2+H_3$

Fig. 7 shows how we represent DA architecture for 3 inputs and 3 filter coefficients where 2^3 combinations are stored in an LUT and based on LSB bits in registers coming as input to LUT particular combination is selected and partial products so obtained are added.

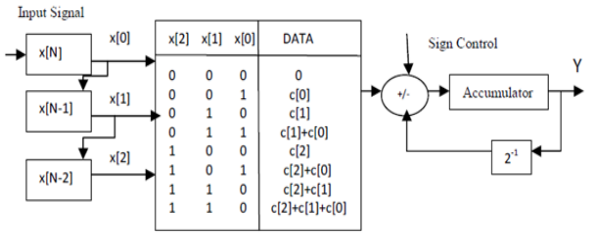

Fig. 7. 2^3 x B LUT based DA FIR filter.

In the DA architecture all the possible binary combinations of the filter coefficients are stored in a memory or lookup table [5], [9]. It is evident that for large values of L, the size of the memory containing the pre computed terms grows exponentially too large to be practical. The memory size can be reduced by dividing the single large memory (2Lwords) into m multiple smaller sized memories each of size 2k where L= m × k.

D. Challenges involved in implementation of Filter

- Representation of filter coefficients
- Optimized realization in terms of hardware

Implementation of half band low pass filter is initially done using Matlab and then by using HDL coding we realize the hardware. Let us consider Matlab implementation for filter, filter coefficients obtained are between 0 and 1 (i.e. for example- 0.1781, -0.9673 etc) and consists of positive as well as negative values so we cannot directly use those values in HDL's. For simplified implementation we realize by converting them to integer values with the help of scaling with required factor in terms of powers of 2. Table II represent the coefficients used in filter realization which are obtained by simulation in Matlab. Filter coefficients are between 0 to 1 so for programming purpose the coefficients shown below are scaled with a 2^4 factor.

TABLE II.
Look up table (LUT) for a 4tap filter with all possible combinations where h denotes filter coefficients

Input coefficients (x)	**Filter coefficients(h)**	**1st stage filtered outputs**	**2nd stage filtered outputs**
39	0	0	0
19	4	156	0
8	8	388	624
5	4	340	2800
-33	0	160	5088

IV. VHDL SIMULATION RESULTS-

A 4tap half-band FIR low pass filter with a cut-off frequency of 8 KHz is realized usingVHDL [9], [10] and Fig. 8 and Fig. 9 shows simulation results for single stage and cascaded filter. The half-band filter is realized in Xilinx 9.2i version target as a 9500XL (Xa9500XL) FPGA device. ISE design software offers a complete design suit based programmable logic devices on Xilinx ISE.

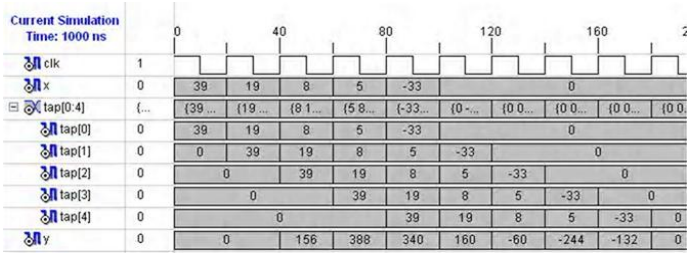

Fig. 8. VHDL simulation result for 4tap half-band filter.

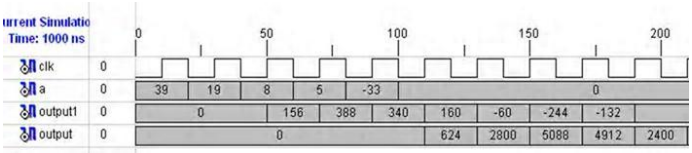

Fig. 9. VHDL simulation result for 4tap cascaded half-band filter (2 stage) .

V. COMPARISON OF OF VARIOUS FACTORS FOR DIFFERENT REALIZATIONS OF 4 TAP HALF BAND FIR FILTER

The Table III clearly shows the delay as well as clock frequency for different realizations. The delay for DAA realization is less comparative to that of other realizations but at the cost of more hardware utilization. In case of transposed form the delay is less than direct form but if there is a constraint on hardware utilization then it is preferable to use transposed form than other two forms else it there is no constraint on memory utilization at the advantage of less delay choosing DAA realization is appropriate.

TABLE III.
Comparison of critical path delay and clock frequency for different 4 tap half-band filter realizations.

Parameter	Direct form	Transposed form	Distributed arithmetic architecture (DAA)
Delay	2.217 ns	1.739 ns	1.666 ns
Clock frequency	450.97 MHz	575.10 MHz	600.96 MHz

TABLE IV.
Hardware utilization for direct form realization.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	35	126800	0%
Number of Slice LUTs	28	63400	0%
Number of fully used LUT-FF pairs	26	37	70%
Number of bonded IOBs	21	210	10%
Number of BUFG/BUFGCTRLs	1	32	3%

TABLE V.
Hardware utilization for transposed form realization.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	35	126800	0%
Number of Slice LUTs	27	63400	0%
Number of fully used LUT-FF pairs	27	35	77%
Number of bonded IOBs	21	210	10%
Number of BUFG/BUFGCTRLs	1	32	3%

TABLE VI.
Hardware utilization for DAA realization.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	131	126800	0%
Number of Slice LUTs	115	63400	0%
Number of fully used LUT-FF pairs	74	172	43%
Number of bonded IOBs	22	210	10%
Number of BUFG/BUFGCTRLs	2	32	6%

The Table IV, Table V, Table VI clearly depicts what percentage of hardware is utilized for different realizations by VHDL simulation in Xilinx 9.2i version. The comparison shows that LUT utilization for DAA is more than direct form and transposed form realizations. Results show that for a real time application where memory is not a constraint and delay should be less the most practical solution is DAA, but delay is considerable along with hardware better to go with the Transposed form of realization.

ACKNOWLEDGMENT

The work was supported by my internal guide Dr.K.Manjunatha Chari, Professor, H.O.D, E.C.E Department, GITAM University, Hyderabad, Telangana, India and external guide U.Naresh Kumar, Scientist-E, R.C.I., Hyderabad, Telangana. India.

REFERENCES

[1] Alan N.Willson and H.J.Orchard, “A design method for half-band FIR filters,” IEEE Trans. Fundamental theory and applications. VOL. 45, NO. 1, Jan 1999.
[2] Kishore A. Kotteri, Amy E. Bell and Joan E. Carletta, “Multiplierless filter bank design: that improve both hardware and image compression performance,” IEEE Trans. Circuits and Systems for vedio tech. VOL. 16, NO. 6, Dec 2006.
[3] Pavel Zahradnik and Miroslav Vlcek, “Equiripple approximation of half- band FIR filters,” IEEE Trans. Circuits and Systems. VOL. 56, NO. 12, Dec 2009.
[4] Alan N.Willson, “Desensitized half-band filters,” IEEE Trans.Circuits and Systems. VOL. 57, NO. 1, Jan 2010.
[5] Narendra Singh Pal, Harjit Pal Singh, R.K.Sarin and Sarabjeet Singh, ”Implementation of high speed FIR filter using serial and parallel distributed arithmetic algorithm,” International Journal of Computer Applications. VOL 25– No.7, July 2011.
[6] Ramesh .R and Nathiya .R,” Realization of FIR filter using modified distributed arithmetic architecture,” Signal & Image Processing: An International Journal (SIPIJ) Vol.3, No.1, and Feb 2012.
[7] M. Yazhini and R. Ramesh,”FIR filter Implementation using modified distributed arithmetic architecture,” Indian Journal of Science and Technology .Vol 6 (5), May 2013.

[8] Abul Fazal Reyas Sarwar1 and Saifur Rahman2, "Design of multiplier Less 32 tap FIR filter using VHDL,"International Open Access Journal of Modern Engineering Research.Vol. 4, Iss. 6, June. 2014.

[9] Sungwook Yu and Swartziander E E. DCT implementation with distributed arithmetic[J]. IEEE Transactions on Computers, 2001,50(9):985~991.

[10] VHDL programming by J.Baskar.