IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 9, Issue 4, Ser. I (Jul. - Aug. 2019), PP 28-37
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Implementation of DAUBECHIES Discrete Wavelet Filter Banks Using Xilinx FPGAS

Ranganadh Narayanam[1]
[1]*Assistant Professor, ECE department, Faculty of Science and Technology,The ICFAI Foundation for Higher Education, Hyderabad, (Deemed to be University under section 3 of UGC Act, 1956), India*
Corresponding Author: Ranganadh Narayanam1

Abstract: *In the recent decade discrete wavelets have become very popular in Digital Signal Processing area due to their capability of simultaneous representation of time-frequency information of a signal. Wavelet transform is especially very important for processing non-stationary signals. Due to the high demand for real time DSP hardware architectures the demand for high throughput and low power in portable devices requires discrete wavelet hardware implementation also to be efficient. In this research three different advanced polyphase architectures for wavelet filter banking are implemented. Xilinx Artix-7 FPGAs are used for implementation through Vivado 2015.2 design suite. The programming is done using Verilog Hardware Description Language (HDL).*

Keywords: *Discrete Wavelets, Wavelet Filter Banks, FPGA, Verilog HDL*

Date of Submission: 04-09-2019 Date of acceptance: 19-09-2019

I. Introduction

Fast Fourier Transform (FFT) is highly useful in analyzing the signals whose frequency content does not change with time. But it does not represent time based frequency analysis; such a barrier can be filled by wavelets. Wavelets are useful for time-frequency analysis. Non stationary signals such as EEG signals, Speech signals require time-frequency analysis that is where wavelets are highly useful in time-frequency analysis than FFT frequency analysis. Due to better resolution and compression capabilities of Wavelets, these days Wavelets are becoming more popular in various applications of Bio-medical signal processing, Speech signal Processing. Due to the increasing applications of Wavelets in speech, image, audio and video it is essential to concentrate on DWT hardware implementation techniques. These hardware implementations to be efficient in terms of power, area, speed, throughput the design has to depend on pipelining, distributed arithmetic, hence efficient memory storage and global data transfer are important criteria for implementing for such applications. In this research we have considered usage of FPGAs to implement these designs due to its capability of reprogramming/reconfigurable and its market for DSP is very emerging. By using high-density field programmable logic devices (FPGAs), the discrete wavelet transformation obtains higher processing speed and lower costs [5] than other implementation methods.

When it comes to non-stationary signal analysis, along with Discrete Wavelet Transform (DWT), Short Time Fourier Transform (STFT) is also one of the important techniques. STFT gives constant resolution at all frequencies which is a drawback and can be overcome by Wavelet Transform which uses multi-resolution technique in which different frequencies are analyzed with different resolutions. In DWT the width of wavelet function changes with each spectral component, unlike the constant width of STFT window. Due to the multi resolution capability the DWT hardware implementation need has become more important for advancing of the technologies [15]. Discrete form of the Wavelet Transform makes the computation easier, makes it easy to implement on hardware and reduces the required resources and computation time.

1.1 wavelet families

Wavelets are mathematical functions that cut up data into different frequency components, and then study each component with a resolution matched to its scale. These basis functions are short waves with limited duration, thus the name "wavelets" is used. The basis functions of the Wavelet Transform are scaled with respect to frequency. There are many different wavelets that can be used as basis (wavelet) functions; a few of them are discussed here. Wavelets are categorized into orthogonal and bi-orthogonal wavelets [4]. There are various wavelet families are available such as Haar, Mortlet, Mexcian Hat, Meyer, Daubechies (db), Coiflet, Symlet.

The following are the wavelet functions for various different wavelets.

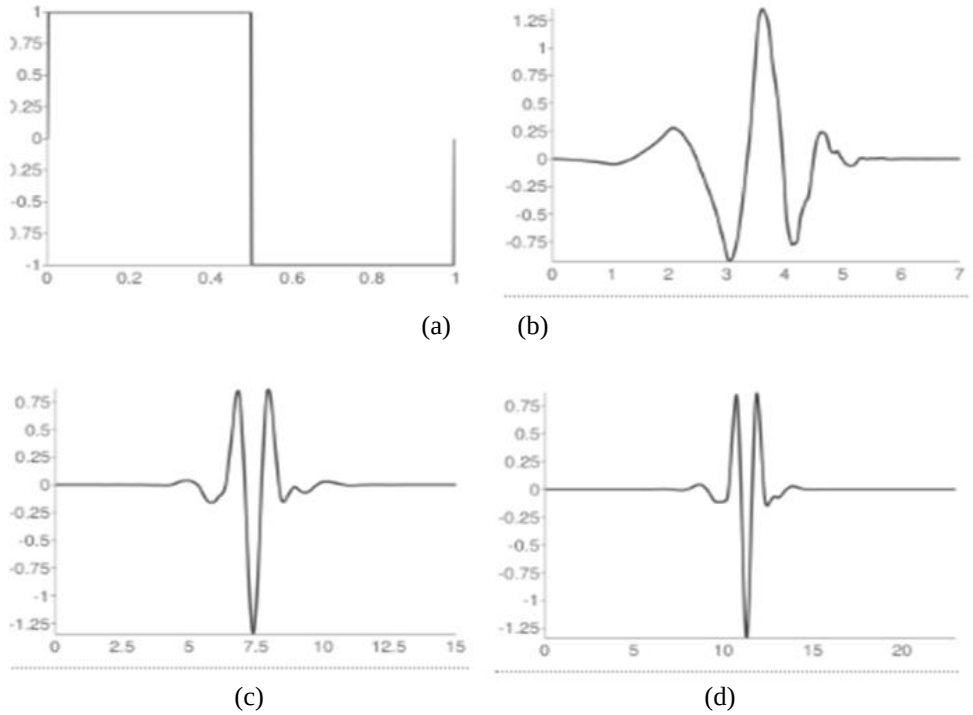

Figure 1 (a) Haar Wavelet (b) Daubechies4 – here the number 4 represents number of vanishing movements (c) Symlet (d) Coiflet.

The first and the simplest is Haar Wavelet. It resembles a step function and it represents same as Daubechies db1. Due to Daubechies wavelets orthonormal wavelets makes discrete wavelet analysis more practicable. Mexican Hat wavelet is proportional to the second derivative function of the Gaussian probability density function [4]. The wavelet is a special case of a larger family of derivative of Gaussian (DOG) wavelets. It is also known as the Ricker wavelet. There is no scaling function associated with this wavelet. In mortlet wavelet both real valued and complex valued wavelets exist. This wavelet has no scaling function, but is explicit. The symlets are nearly symmetrical wavelets proposed by Daubechies as modifications to the db family [4]. The properties of the two wavelet families are similar. The Meyer wavelet and scaling function are defined in the frequency domain. The basis function of any wavelet produces all wavelet functions used in the transformation through translation and scaling it determines the characteristics of the resulting wavelet transform. Hence for the chosen wavelet transform has to be effective, we have to consider most importantly the details of the application and proper mother wavelet. Haar, Symlet, Daubechies and coiflet wavlets are compactly supported orthogonal wavelets, and hence when combined with Meyer wavelets can be used for perfect reconstruction [4]. Haar, Daubechies, Meyer wavelets are asymmetric, orthogonal and bio-orthogonal. But symlet, coiflets wavelets are near symmetric, but orthogonal and bi-orthogonal.

1.2 Applications

One of the most prominent applications of wavelet transform is in FBI finger print compression standard. To store in the data bank of finger print wavelet transform is used to compress the finger print pictures. At high compression ratios, DCT faces the problem of severe blocking problem after reconstruction, which is overcome using Discrete Wavelet Transform. Due to the reason that most prominent information is located in high amplitudes and less prominent information stored at low amplitudes so, wavelets are useful at eliminating low amplitude components, and then compress the data with high compression ratio. DWT is highly chosen in JPEG 2000 compression standard. To reduce mobile data transmission time wavelets are used in speech compression. Wavelet transforms are useful in feature extraction, speech recognition, de-noising, edge detection, echo cancellation, audio and video compression applications, In OFDM, Biomedical imaging. The ability of DWT to reduce distortion after reconstruction it is getting high popularity.

II. Architectures For Wavelet Filter Banking

The following 4 are very important filter banking structures

(1)Direct form structure (2) Polyphase structure (3) Lattice structure (4) Lifting structure

For hardware implementation efficiency and accuracy of computing of DWT depends on the structure chosen. Each structure has its own advantages and drawbacks, but basing on the application its efficiency varies. So basing on the application properly chosen structure gives most suitable implementation can be done. Direct form found to be more inefficient for DWT and almost never be used [14, 15]. Lattice and lifting structure implementations found to be more efficient in terms of number of computations than polyphase structure. By incorporating Distributed Arithmetic in the Polyphase implementations, it can be used more efficiently in case of long filters and highly efficient than lattice and lifting structures [12, 14, 15]. Lattice structure puts constraint on length of the filters in case of most of the linear phase filters. In case of lattice and lifting schemes the output depends on the previous filtering units, the filtering units cannot work in parallel. But in case of polyphase structure the units can work in parallel so to get result in less time [12, 13, 15]. Hence the pipelining can be used. So, keeping in mind of advantages of Polyphase structure in hardware implementations in this research we have developed various architectures for Polyphase structure.

2.1 Polyphase Structure

If the output consists of N samples due to decimation by 2 we use only N/2 samples in case of direct form structure analysis filter bank. So computation of remaining N/2 samples becomes redundant. The polyphase structure takes the advantage of the fact that the input signal is split into odd and even samples, which automatically decimates the input by 2, similarly the filter coefficients are also split into even and odd components so that the signal even samples X_{even} convolves with low pass even part of the filter $G0_{even}$ and input signal odd samples X_{odd} convolves with $G0_{odd}$ of the filter. The two phases are added together in the end to produce the low pass output. Similar lines are applied to the high pass filter H_{even} and H_{odd}. The matrix representation Polyphase analysis operation is given in equation (1).

$$\begin{bmatrix} G_{0\,even} & G_{0\,odd} \\ H_{0\,even} & H_{0\,odd} \end{bmatrix} \times \begin{bmatrix} X_{even} \\ z^{-1}X_{odd} \end{bmatrix} = H_p \begin{bmatrix} X_{even} \\ z^{-1}X_{odd} \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \quad (1)$$

The length of G_0 filter is twice as long as it's even and odd filters, as they are obtained by splitting G_0. The evn and odd terms are filtered separately by the even and odd coefficients of the filters, the filters can operate in parallel, which improve the efficiency [1, 4]. The polyphase analysis and synthesis filter banks are given in the figure 2. In the direct from synthesis filter bank, input is first samples by adding zeros and then filtered. In case of polyphase synthesis filter bank, the filters come first followed by up-samplers which again reduce the number of computations in the filtering operations by half. So, in case of both analysis and synthesis filter banks the number of computations reduces by 50%. Hence the polyphase filter banks give efficient hardware realizations.

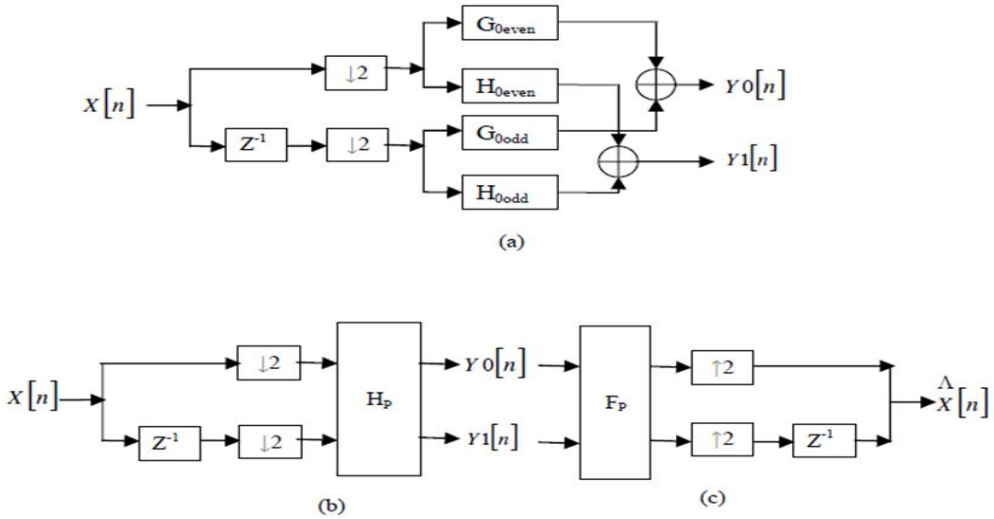

Figure 2 Polyphase structure of (a) Analysis filter bank (b) Equivalent representation of synthesis filter bank. (c) Synthesis filter bank.

2.2 Filter Banking using Distributed Arithmetic (DA) Technique

2.2.1 Distributed-Arithmetic based approach

If filter coefficients are known in advance, the DA technique is very important [1, 4] for computing dot product equations such as the ones in DSP FPGA applications. In this DA technique to replace the multipliers which occupy large area with small tables for pre-computed sums stored on FPGA Look-Up Tables. The inner sum equation is rearranged so that the MAC operation is reduced to a series of LUT calls, and 2's complement shifts and adds [13, 14, 15]. For example for a 8-tap filter, the conventional MAC operation, and the DA-based shift-add operation are given in the following **Figure 3 (a) & (b).** In case of DA architecture, the input samples are fed to the parallel-to-serial shift register cascade. For a filter of N-tap, considering B-bit input samples, it contains N shift register each of B-bits. As the input samples are shifted serially through the B-shift registers, the one bit outputs from each of N registers of the shift register cascade work as address inputs for the LUT. The LUT accepts the N bit input vector x_b, c[n] represents the N-tap constant coefficient filter and outputs the value of $\sum_{n=0}^{N-1} c[n].x_b[n]$, which is already stored in the LUT. 2^N word LUT is required for an N-tap filter. The LUT output is then shifted based on the weight of x_b and then accumulated. For each bit of the input sample this process is followed, before a new output sample is available. Every B clock cycles, a new inner product y is computed for a B-bit input precision.

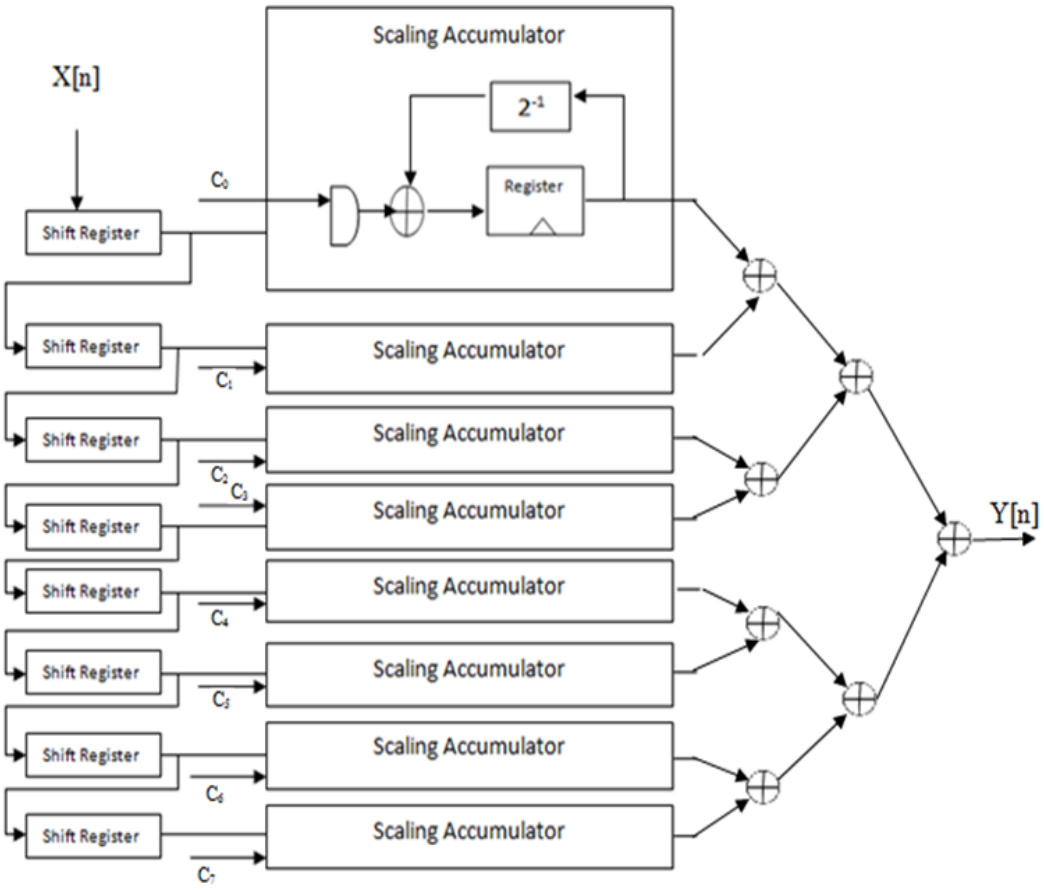

Figure 3 (a) Conventional MAC

Here a 8-tap serial FIR filter is considered, its coefficients are C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7. The DA LUT **Table 1**, which consists of the sums of products of the N bit (here it is N = 8) input vector X_b and all the possible combinations of filter coefficients.

Conventional MAC designs of FIR filters the performance in terms of throughput depends on filter length and inversely proportional to its length (no of TAPs). In case of DA based designs it is independent of length, but dependent on input bit precision. If filter size increases the number of hardware resources increases. When the filter length is increased, the throughput remains the same while the logic resources increase. If long filters, instead of creating a long table, smaller tables are created by partitioning, and outputs can be combined.

For the above specified filters, Polyphase form can be obtained by splitting the filters and the input, x[n] into even and odd phases to obtain four different filters, hence the length of the filter is halved and smaller LUTs are enough.

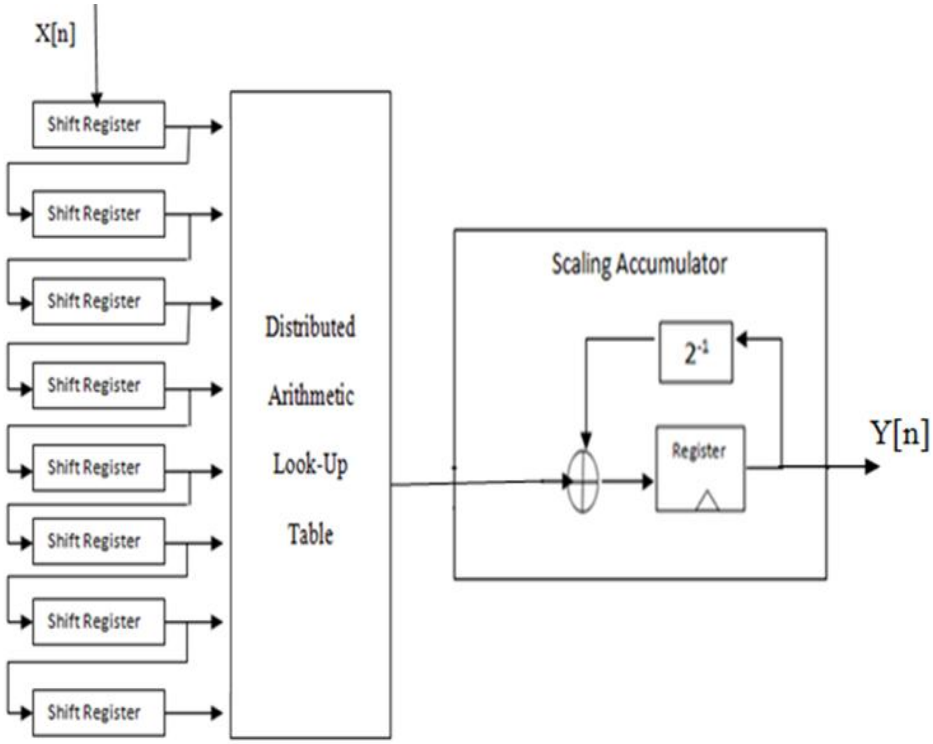

Figure 3 (b) Distributed Arithmetic Architecture
Figure 3 (a) & (b) for 8-Tap filter

Address	Data
00000000	**0**
00000001	**C_0**
00000010	**C_1**
00000011	**C_0+C_1**
……	**…….**
……	**…….**
11111110	**$C_7+C_6+C_5+C_4+C_3+C_2+C_1$**
11111111	**$C_7+C_6+C_5+C_4+C_3+C_2+C_1+C_0$**

Table 1 DA-LUT table for a 8-tap filter

Speed Improvement using Parallel DA.

DA-Based design is bit-serial approach. So for processing B-bit input, it takes B-clock cycles for one bit output result. So such serial distributed arithmetic do have low throughput. If we partition the input words into smaller input words and processing them in parallel, increase the parallelism, throughput increases hence speed increases and required LUTs increases. If we partition the input B-bits into M sub words, then every new input comes out for each B/M clock cycles. In this type of fully parallel DA filters design several bits of the input are processed in a clock cycle and achieve maximum speed. At the expense of increased FPGA resources, this design style provides high performance. Example PDA architecture is given in the following figure for 8-bit input with 4-tap filter. This is given in **Figure 4**.

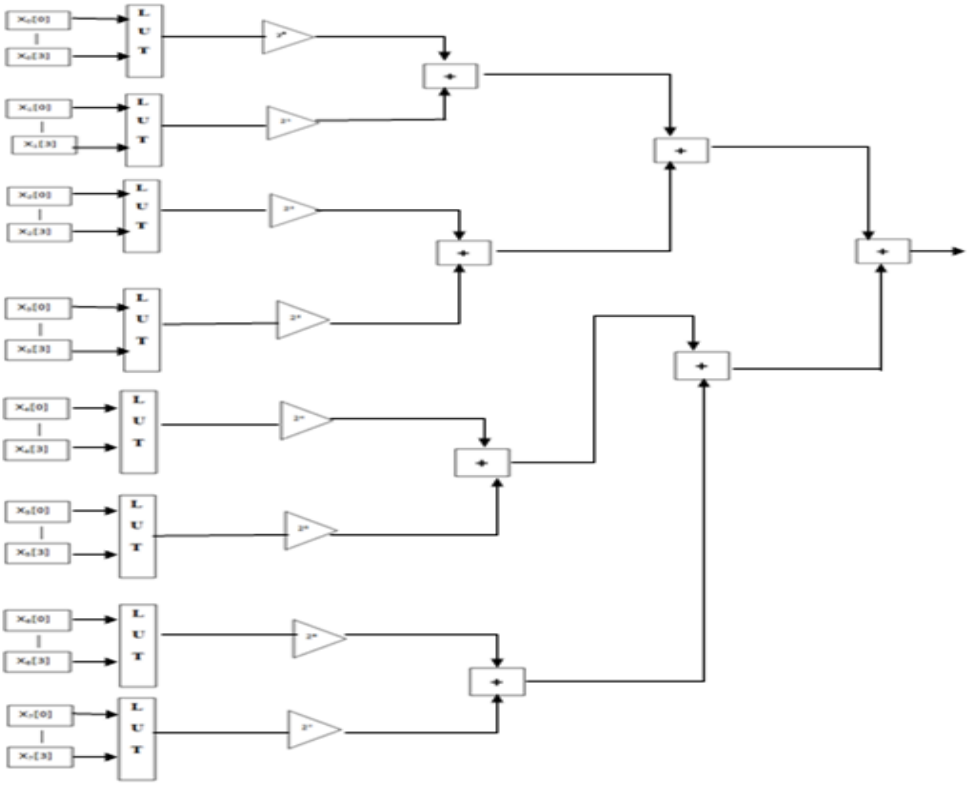

Figure 4 Parallel Distributed Arithmetic Architecture. The inputs for the first LUT are $x_0[0]$ to $x_0[3]$; for the second LUT are x1[0] to x1[3], and so on for eighth LUT are $x_7[0]$ to $x_7[3]$. In the buffers triangles the values are 2^0, 2^1 and so on to 2^7.

2.2.2 Coefficient over Input Distribution (CID) approach based on DA for the filter bank

A novel architecture is proposed in [10], in which coefficient matrix is distributed over the input. In the previous architecture, LUT size exponentially increases with increase in input precision, and this increases amount of logic resources required. The advantage of this current architecture over the previous one is that, in this we do not require any memory or LUT tables. Logic resources consumption reduces tremendously. With this architecture, the inner mathematical equation can be implemented using set of unique adder system based on the coefficient bits consisting of zeros and ones. Based on coefficient bit weight, the output, y, can be computed by shifting and accumulating the results of adder system accordingly. So the whole system can be implemented just by using adders and shifters.

III. Implementation Of Filter Bank Architectures

In this research the architectures for wavelet filter banking are implemented for poly-phase structure. The above specified architectures

1) Polyphase architecture
2) Distributed Arithmetic based Parallel architecture
3) CID architectures: A modified DA architecture

are implemented on Field Programmable Gate Arrays. The FPGAs utilized are Xilinx Artix-7 FPGAs, and the Verilog HDL programming is done. The 8 Tap Daubechies wavelets orthogonal filter banks implemented. Only the analysis filter banks are considered, synthesis filter banks can be designed using the similar approaches. All the coefficients are real numbers, so we convert all of them into integers to get the required size integer and required number of digits precision and then convert those integers into binary. Then when the results are obtained proper quantization factor is utilized to divide the results to get back the original ones.

The Dabechies 8-Tap orthogonal filter bank

This filter is widely used due to its orthogonal properties and it satisfies the perfect reconstruction conditions. The coefficients of analysis filters are given in the Table No 2.

Tap	Low Pass Filter	High Pass Filter
0	0.128747426620186	-0.05441584224308161
1	0.00047248457399797254	0.3128715909144659

2	-0.2840155429624281	-0.6756307362980128
3	-0.015829105256023893	0.5853546836548691
4	0.5853546836548691	0.015829105256023893
5	0.6756307362980128	-0.2840155429624281
6	0.3128715909144659	-0.00047248457399797254
7	0.05441584224308161	0.128747426620186

Table 2 Low pass and High pass filter coefficients of 8-Tap Daubechies filter

Floating point operations are slower than integer operations and take much hardware resources, and consume high power when it comes to hardware design when compared to integer operations. So, for ease of hardware implementation, floating point coefficients are shifted by 2^{13} and the corresponding integer values are used. This gives good precision of 5 decimal places. To get the correct result, the output is shifted by 2^{-13} to get the correct result.

IV. . Implementation

Daubechies orthogonal wavelet filter banks implementation of 8-Tap filter bank is explained in this section with all the three architectures described. The **Architecture 1:** Polyphase filter bank implementations architecture; **Architecture 2:** Distributed Arithmetic based architectures: Parallel DA architectures & **Architecture 3:** CID architectures and implementation results are discussed.

Architecture 1: Polyphase Implementation:

In this each of the filter of the analysis bank is divided into even and odd phase to obtain the 4-tap filters. The table is given in Table No 3**.** The incoming input samples, x[n], are multiplexed into even sample (Xeven) and odd samples (Xodd). Convolution of even samples with even filter coefficients and the odd samples with a delay are convolved with odd filter coefficients. At the end the even and odd phases are added. The approximation or scaling coefficients are represented in Low pass filter output, and the detail or wavelet coefficients are represented in high pass filter output.

Low Pass Filter		High Pass Filter	
Even LPF	Odd LPF	Even HPF	Odd HPF
0.128747426620186	0.00047248457399797254	-0.05441584224308161	0.3128715909144659
-0.2840155429624281	-0.015829105256023893	-0.6756307362980128	0.5853546836548691
0.5853546836548691	0.6756307362980128	0.015829105256023893	-0.2840155429624281
0.3128715909144659	0.05441584224308161	-0.00047248457399797254	0.128747426620186

Table 3 Polyphase filters for 8 Tap Daubechies Discrete Wavelet filter bank

Architecture 2: Parallel DA-based implementation

This is highly suitable for LUT based architectures of FPGAs. It can be observed from the Table 3 that there are 4, 4-tap filters. Each 4 tap filter needs 16 input LUT. Low pass even coefficients are same as High pass odd coefficients, but mirror images. Similarly, low pass odd coefficients are same as high pass even coefficients with just a difference of sign. So instead of having 4 different LUTs we can use just 2 LUTs and share them for all filters with multiplexing process, which causes reduction in speed but at less hardware resource utilization. Here it is 13 bit precision is being used, so Serial Distributed Arithmetic based implementation causes each output takes 13 clock cycles, which is not always suitable in real time applications. So, parallel Distributed Arithmetic based architecture described above is highly useful. All 13 input bits are processed at a time and this requires 13 identical LUTs are required for each filter. Here speed increases at the expense of high hardware resource utilization.

Architecture 3: Coefficient over Input Distribution (CID) approach based on DA for the filter bank: a modified DA Implementation

In this architecture adder trees are used instead of LUTs. Similar to LUTs data, here also trees are formed by filter coefficients. The input samples are passed through a delay line so that the adder tree gets the data in parallel. This is exemplified through the figure 5(a). Low pass even filter can choose X_0, X_2; X_4, X_6 and so on from the adder tree as inputs to the adder specified in figure 5(b). High pass odd filter can choose X_1, X_3; X_5, X_7 and so on from the adder tree as inputs to the adder specified in figure (b). As Low pass odd filter and High pass even filter have the same coefficients so no need of additional adders to these coefficients. So, here in this case we can have only 1 adder for all the results. But if we want to do the addition process in parallel we can have more adders which work in parallel.

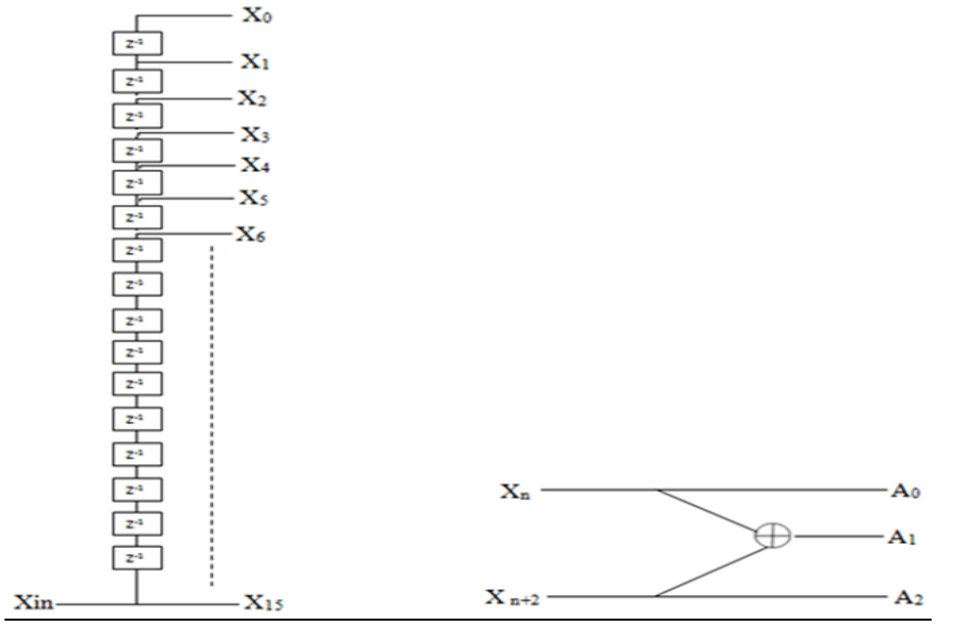

Figure 5 Polyphase implementation of 8 Tap Daubechies with CID modified DA approach (a) Delay line (b) adder tree

Adder tree gets output results after execution. Then those outputs are scaled according to the respective bit-weight and then added to get the final filter outputs. A pipelined adder exemplified in the figure 6 can be used for this process. To obtain the final outputs, even and odd phases are added, which is the simplest process for the whole design to be implemented with minimal number of shift and add operations.

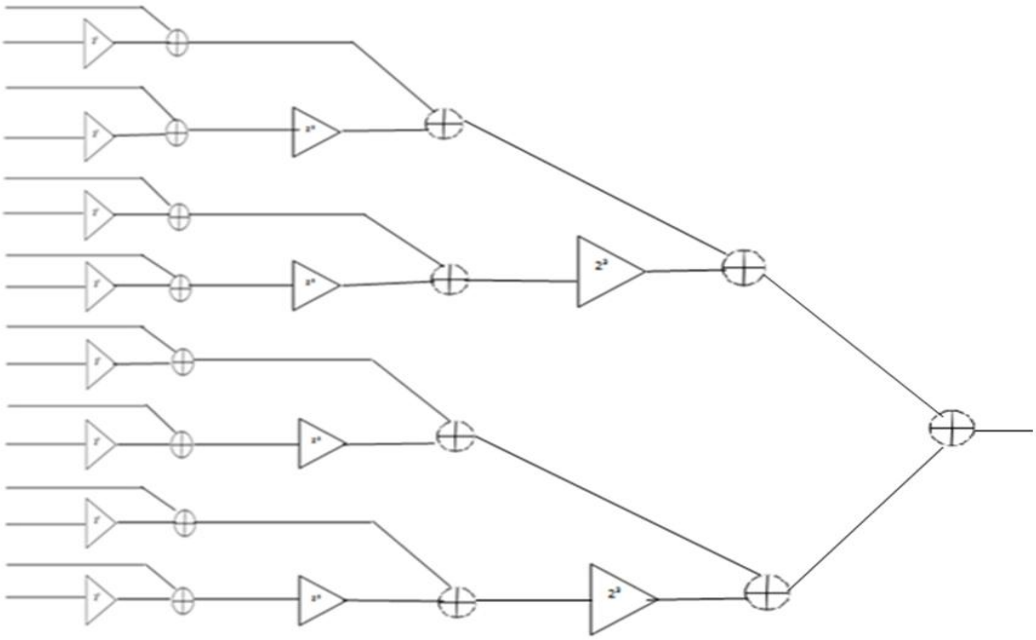

Figure 6 pipe lined adder shifter for 8-tap Daubechies wavelet filter bank. The first stage buffers have 2^1 in the triangles; second stage buffers have 2^2 in the triangles; and third stage buffers have 2^3 in the triangles.

V. Implementation Results

In this research the hardware implementation of 8-Tap Daubechies wavelet filter banking is done using 3 different techniques. All the three above described architectures are implemented using Vivado 2015.2 synthesis tool, Verilog HDL programming on Xilinx Artix-7 FPGAs. The performances of all the three architectures are compared in terms of clock speed, power utilization, and hardware resource utilization. The details are given in the following table 4.

Architecture 1 corresponds to Polyphase Implementation; Architecture 2 corresponds to Parallel DA-based implementation; Architecture 3 corresponds to Coefficient over Input Distribution (CID) approach based on DA for the filter bank: a modified DA approach.

From the hardware resource utilization results it is observable that Architecture 2 takes much hardware than all of them. Architecture 2 takes 41% more resources than Architecture 1; and 50.5% more resources than Architecture 3. The architecture 2 takes more resources because it utilizes fully parallel architecture of DA based approach. It would have consumed less hardware if not parallel architecture, but if not parallel architecture then less throughput. So, if area is a constraint Architecture 3 would be a good choice.

One more performance consideration is clock frequency. The table 4 and bar graphs Figures 7, 8, 9 show the maximum operating frequency of 3 architectures. From the obtained results it is observed that Architecture 2 is much faster than all architectures. It works 35.36% much faster than Architecture 1, and 51.82% much faster than Architecture 3. So, Architecture 2 is a best choice if we require high speed of operation and operating at higher sampling rates.

From the power utilization results it is clear that Architecture 3 consumes less power when compared to all the architectures. So, for power and area efficiency Architecture 3 is a best choice; and for high speed and high throughput Architecture 2 is a best choice.

	FFs (267600)	LUTs (133800)	Total hardware (FFs+LUTs)	On chip power (mW)	Clock speed (MHz)	Time period (ms)
Architecture-1	856	530	1386	210.2	60.89	16.423
Architecture-2	1128	826	1954	440.5	82.42	12.133
Architecture-3	798	500	1298	187.4	54.29	18.420

Table 4 the performance of all the three implemented architectures

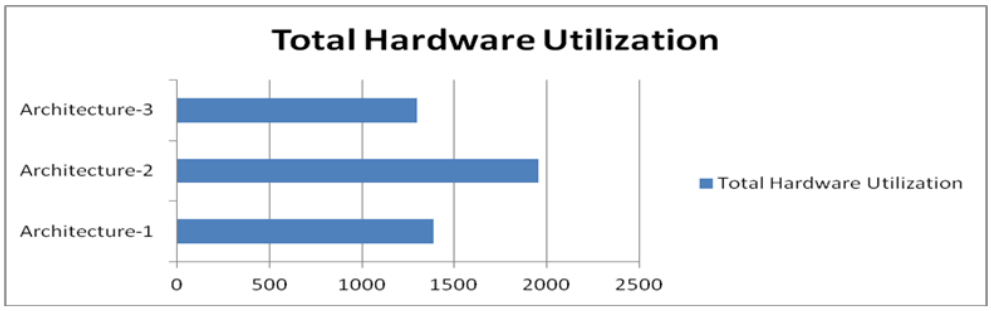

Figure 7 Utilization of FPGA Hardware resources

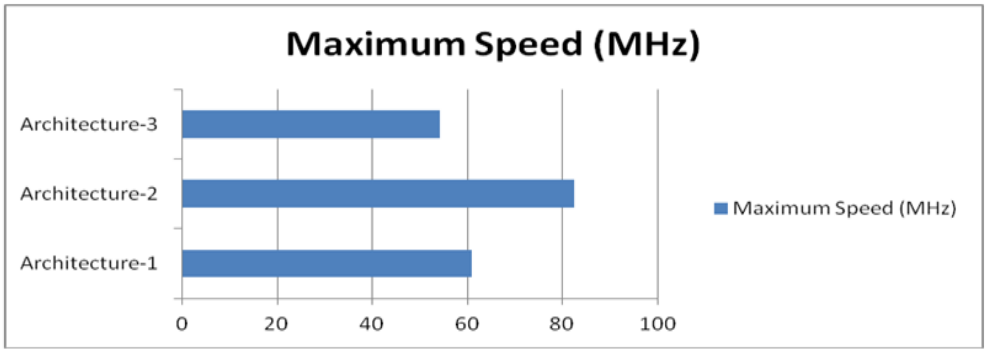

Figure 8 Maximum speed (MHz)

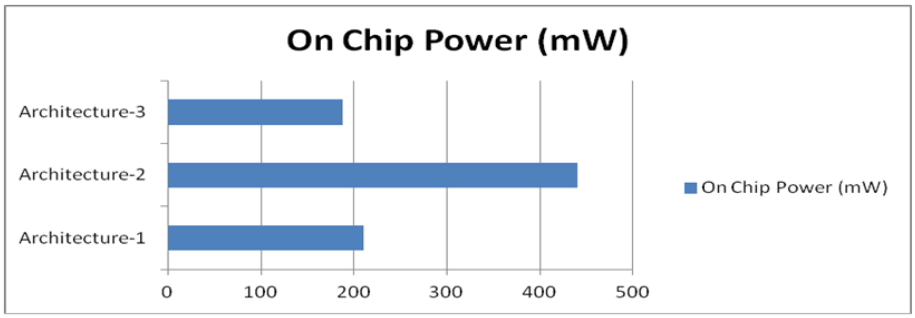

Figure 9 Total onchip power utilization (mW)

VI. Conclusion

For multi-resolution analysis of signals, Discrete Wavelet transforms are highly useful. This transform can be implemented using Wavelet filter banks. In this research three advanced architectures have been implemented for 8 Tap Daubechies wavelets: Polyphase architecture, Polyphase with fully parallel DA architecture, Polyphase with shifting and adding based modified DA architecture. All the three architectures are implemented on Xilinx Artix-7 FPGAs using Vivado synthesis tool, and Verilog HDL. Performances of the three architectures are analyzed using clock speed, on-chip power, hardware resource utilization (area). The Architecture 2 is found to be best in case of high clock speed, higher sampling rates of operation and high throughput. The architecture 3 is found to be best choice in terms of low power and low area.

References

[1]. Husam Y.A, B.Berk.U, "An optimized two-level discrete wavelet implementation using residue number system", EURASIP journal of advances in signal processing, 2018.
[2]. Mehboob alam, W.badawy, Rahman, "Efficient distributed arithmetic based DWT architecture for multimedia applications", Conference: System-on-Chip for Real-Time Applications, 2013. Proceedings. The 3rd IEEE International Workshop
[3]. Reena Rajan, Lijesh L, "High Speed DA based Discrete Wavelet Transform Digital Design for Image Processing using Verilog", IJERT – 2017.
[4]. Gilbert Strang, T Ngyuen, "Wavelets and Filter banks", wellesley-cambridge press, 1997.
[5]. Chen Jing,Hou Yuan Bin, "Efficient Wavelet Transform on FPGA Using Advanced Distributed Arithmetic",2007 8th International Conference on Electronic Measurement and Instruments, IEEE,2007.
[6]. UM Baese, "Dsigital Signal Processing using Field Programmable Gate Arrays", springer-verlag, 2001.
[7]. J.Ramirez, A.Garcia, "design of RNS based Distributed Arithmetic DWT filter banks", 2001 IEEE international conference on Accoustics, speech, and sognal processing,2001.
[8]. Husam A, Burak B "A Comparative Performance of Discrete Wavelet Transform Implementations Using Multiplierless", Chapers, Intechopen - Wavelets theory and applications, 2018.
[9]. Shruti S.Velukar ,"FPGA Implementation of Fir Filter using Distributed Arithmetic Architecture for DWT", International Journal of Computer Applications, 2014.
[10]. M.Alam, C.A.Rahman, "Efficient Distributed Arithmetic based DWT architecture for multimedia applications, proceedings of the 3rd IEEE international workshop on system-on chip for real-time applications, june 2003.
[11]. Husam ALZAQ , Burak Berk "A comparative analysis of 1-level multiplier-free discrete wavelet transform implementations on FPGAs", Turkish Journal of Electrical Engineering & Computer Sciences, 2018.
[12]. Xilinx Incorporation, "The role of Distributed Arithmetic in FPGA based signal processing, xilinx application notes, San Jose, CA.
[13]. Ahmed Saeed ; Hani Fikri Ragai, "Implementation of fast discrete wavelet transform for vibration analysis on an FPGA", 2012 8th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP),IEEE, 2012.
[14]. Noor Huda Ja'afar ; Afandi Ahmad ; Abbes Amira, "Distributed arithmetic architecture of Discrete Wavelet Transform (DWT) with hybrid method", 2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS), IEEE, 2013.
[15]. D.Sripathi; S.Y. Foo,"Efficient implementations of discrete wavelet transforms using FPGAs",WctLbook, 2003.
[16]. Koushik Roy, S.C.prasad,"Low-power CMOS VLSI circuit design", John Wisely, 2000.

Ranganadh Narayanam " Implementation of DAUBECHIES Discrete Wavelet Filter Banks Using Xilinx FPGAS. " IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) , vol. 9, no. 4, 2019, pp. 28-37.