Contents

List Of Figures

List Of Tables

ABSTRACT

Recently the Wavelet Transform has gained a lot of popularity in the field of signal processing. This is due to its capability of providing both time and frequency information simultaneously, hence giving a time-frequency representation of the signal. The traditional Fourier Transform can only provide spectral information about a signal. Moreover, the Fourier method only works for stationary signals. In many real world applications, the signals are non-stationary. One solution for processing non-stationary signals is the Wavelet Transform.

Currently, there is tremendous focus on the application of Wavelet Transforms for real-time signal processing. This leads to the demand for efficient architectures for the implementation of Wavelet Transforms. Due to the demand for portable devices and realtime applications, the design has to be realized with very low power consumption and a high throughput.

In this report, different architectures for the Discrete Wavelet Transform filter banks are presented. The architectures are implemented using Field Programmable Gate Array devices. Design criteria such as area, throughput and power consumption are examined for each of the architectures so that an optimum architecture can be chosen based on the application requirements. A scalable architecture for the computation of a three-level Discrete Wavelet Transform along with its implementation is presented.

CHAPTER.1

1. INTRODUCTION

1.1. Introduction

In general, signals in their raw form are time-amplitude representations. These time-domain signals are often needed to be transformed into other domains like frequency domain, time-frequency domain, etc., for analysis and processing. Transformation of signals helps in identifying distinct information which might otherwise be hidden in the original signal. Depending on the application, the transformation technique is chosen, and each technique has its advantages and disadvantages.

1.2. Why Wavelet Transforms?

In most Digital Signal Processing (DSP) applications, the frequency content of the signal is very important. The Fourier Transform is probably the most popular transform used to obtain the frequency spectrum of a signal. But the Fourier Transform is only suitable for stationary signals, i.e., signals whose frequency content does not change with time. The Fourier Transform, while it tells how much of each frequency exists in the signal, it does not tell at which time these frequency components occur.

Signals such as image and speech have different characteristics at different time or space, i.e., they are non-stationary. Most of the biological signals too, such as, Electrocardiogram, Electromyography, etc., are non-stationary. To analyze these signals, both frequency and time information are needed simultaneously, i.e., a time-frency representation of the signal is needed.

To solve this problem, the Short-Time Fourier Transform (STFT) was introduced. The major drawback of the STFT is that it uses a fixed window width. The Wavelet Transform, which was developed in the last two decades, provides a better timefrequency representation of the signal than any other existing transforms.

1.3. Short Time Fourier Transform Vs. Wavelet Transform

The STFT is a modified version of the Fourier Transform. The Fourier Transform separates the waveform into a sum of sinusoids of different frequencies and identifies their respective amplitudes. Thus it gives us a frequency-amplitude representation of the signal. In STFT, the non-stationary signal is divided into small portions, which are assumed to be stationary. This is done using a window function of a chosen width, which is shifted and multiplied with the signal to obtain the small stationary signals. The Fourier Transform is then applied to each of these portions to obtain the Short Time Fourier transform of the signal.

The problem with STFT goes back to the Heisenberg uncertainty principle which states that it is impossible for one to obtain which frequencies exist at which time instance, but, one can obtain the frequency bands existing in a time interval. This gives rise to the resolution issue where there is a trade-off between the time resolution and frequency resolution. To assume stationarity, the window is supposed to be narrow, which results in a poor frequency resolution, i.e., it is difficult to know the exact frequency components that exist in the signal; only the band of frequencies that exist is obtained. If the width of the window is increased, frequency resolution improves but time resolution becomes poor, i.e., it is difficult to know what frequencies occur at which time intervals. Also, choosing a wide window may violate the condition of stationarity. Consequently, depending on the application, a compromise on the window size has to be made. Once the window function is decided, the frequency and time resolutions are fixed for all frequencies and all times.

The Wavelet Transform solves the above problem to a certain extent. In contrast to STFT, which uses a single analysis window, the Wavelet Transform uses short windows at high frequencies and long windows at low frequencies. This results in multiresolution analysis by which the signal is analyzed with different resolutions at different frequencies, i.e., both frequency resolution and time resolution vary in the time-frequency plane without violating the Heisenberg inequality.

In Wavelet Transform, as frequency increases, the time resolution increases; likewise, as frequency decreases, the frequency resolution increases. Thus, a certain high frequency component can be located more accurately in time than a low frequency component and a low frequency component can be located more accurately in frequency compared to a high frequency component.

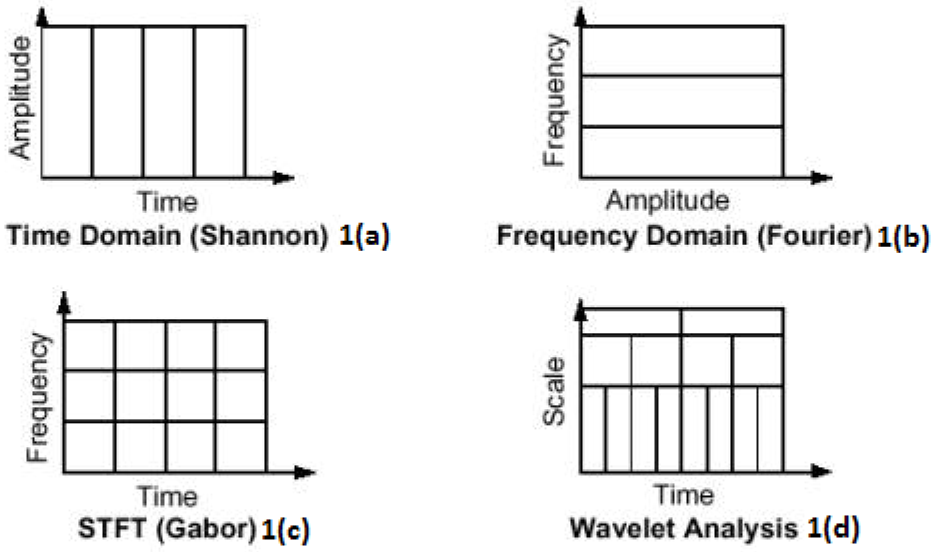


Figure 1*The* Time-Frequency tiling for 1(a) Time-Domain 1(b) Frequency-Domain 1(c) STFT 1(d) DWT.

Figure 1(a) shows the time-frequency tiling in the time-domain plane and figure 1(b) shows the tiling in frequency-domain plane. It is seen that figure 1(a) does not give any frequency information and figure 1(b) does not give any time information. Similarly figure 1(c) shows the tiling in STFT and figure 1(d) shows the tiling in Wavelet Transform. It is seen that STFT gives a fixed resolution at all times, whereas Wavelet Transform gives a variable resolution.

The Wavelet Transform was developed independently in applied mathematics and signal processing. It is gradually substituting other transforms in some signal processing applications. For example, previously, the STFT was extensively used in speech signal processing, and Discrete Cosine Transform (DCT) was used for image compression. But now, the Wavelet Transform is substituting these, due to its better resolution properties and high compression capabilities.

1.4. The Need For Efficient Discrete Wavelet Transform Architecture

The properties of Wavelet Transform allow it to be successfully applied to nonstationary signals for analysis and processing, e.g., speech and image processing, data compression, communications, etc. Due to its growing number of applications in various areas, it is necessary to explore the hardware implementation options of the Discrete Wavelet Transform (DWT).

An efficient design should take into account aspects such as area, power consumption, throughput, etc. Techniques such as pipelining, distributed arithmetic, etc., help in achieving these requirements. For most applications such as speech, image, audio and video, the most crucial problems are the memory storage and the global data transfer.

Therefore, the design should be such that these factors are taken into consideration.

CHAPTER.2

2. THE DISCRETE WAVELET TRANSFORM

2.1. Introduction

The transform of a signal is just another form of representing the signal. It does not change the information content present in the signal. The Wavelet Transform provides a time-frequency representation of the signal. It was developed to overcome the short coming of the Short Time Fourier Transform (STFT), which can also be used to analyze non-stationary signals. While STFT gives a constant resolution at all frequencies, the Wavelet Transform uses multi-resolution technique by which different frequencies are analyzed with different resolutions.

A wave is an oscillating function of time or space and is periodic. In contrast, wavelets are localized waves. They have their energy concentrated in time or space and are suited to analysis of transient signals. While Fourier Transform and STFT use waves to analyze signals, the Wavelet Transform uses wavelets of finite energy.

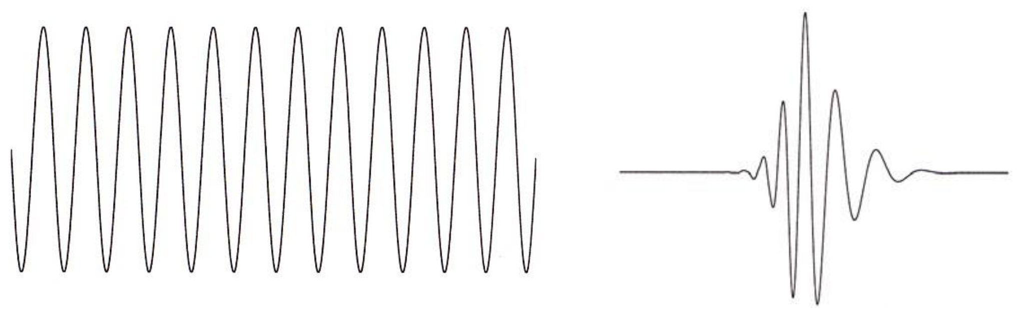


Figure 2 Demonstration of (a) a Wave and (b) a Wavelet [2].

The wavelet analysis is done similar to the STFT analysis. The signal to be analyzed is multiplied with a wavelet function just as it is multiplied with a window function in STFT, and then the transform is computed for each segment generated. However, unlike STFT, in Wavelet Transform, the width of the wavelet function changes with each spectral component. The Wavelet Transform, at high frequencies, gives good time resolution and poor frequency resolution, while at low frequencies, the Wavelet Transform gives good frequency resolution and poor time resolution.

2.2. The Continuous Wavelet Transform And The Wavelet Series

The Continuous Wavelet Transform (CWT) is provided by equation 2.1, where x(t) is the signal to be analyzed. ψ(t) is the mother wavelet or the basis function. All the wavelet functions used in the transformation are derived from the mother wavelet through translation (shifting) and scaling (dilation or compression).

$$X_{WT}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t) \cdot \psi^* \left(\frac{t-\tau}{s} \right) dt \qquad 2.1$$

The mother wavelet used to generate all the basis functions is designed based on some desired characteristics associated with that function. The translation parameter τ relates to the location of the wavelet function as it is shifted through the signal. Thus, it corresponds to the time information in the Wavelet Transform. The scale parameter *s* is defined as |1/frequency| and corresponds to frequency information. Scaling either dilates (expands) or compresses a signal. Large scales (low frequencies) dilate the signal and provide detailed information hidden in the signal, while small scales (high frequencies) compress the signal and provide global information about the signal. Notice that the Wavelet Transform merely performs the convolution operation of the signal and the basis function. The above analysis becomes very useful as in most practical applications, high frequencies (low scales) do not last for a long duration, but instead, appear as short bursts, while low frequencies (high scales) usually last for entire duration of the signal.

The Wavelet Series is obtained by discretizing CWT. This aids in computation of CWT using computers and is obtained by sampling the time-scale plane. The sampling rate can be changed accordingly with scale change without violating the Nyquist criterion. Nyquist criterion states that, the minimum sampling rate that allows reconstruction of the original signal is 2ω radians, where ω is the highest frequency in the signal. Therefore, as the scale goes higher (lower frequencies), the sampling rate can be decreased thus reducing the number of computations.

2.3. The Discrete Wavelet Transform

The Wavelet Series is just a sampled version of CWT and its computation may consume significant amount of time and resources, depending on the resolution required. The Discrete Wavelet Transform (DWT), which is based on sub-band coding is found to yield a fast computation of Wavelet Transform. It is easy to implement and reduces the computation time and resources required. The foundations of DWT go back to 1976 when techniques to decompose discrete time signals were devised [5]. Similar work was done in speech signal coding which was named as sub-band coding. In 1983, a technique similar to sub-band coding was developed which was named pyramidal coding. Later many improvements were made to these coding schemes which resulted in efficient multi-resolution analysis schemes.

In CWT, the signals are analyzed using a set of basis functions which relate to each other by simple scaling and translation. In the case of DWT, a time-scale representation of the digital signal is obtained using digital filtering techniques. The signal to be analyzed is passed through filters with different cutoff frequencies at different scales.

2.4. DWT And Filter Banks

2.4.1 Multi-Resolution Analysis using Filter Banks

Filters are one of the most widely used signal processing functions. Wavelets can be realized by iteration of filters with rescaling. The resolution of the signal, which is a measure of the amount of detail information in the signal, is determined by the filtering operations, and the scale is determined by upsampling and downsampling (subsampling) operations[5].

The DWT is computed by successive lowpass and highpass filtering of the discrete time-domain signal as shown in figure 3. This is called the Mallat algorithm or Mallat-tree decomposition. Its significance is in the manner it connects the continuoustime mutiresolution to discrete-time filters. In the figure, the signal is denoted by the sequence x[n], where n is an integer. The low pass filter is denoted by G_0 while the high pass filter is denoted by H_0. At each level, the high pass filter produces detail information, d[n], while the low pass filter associated with scaling function produces coarse approximations, a[n].

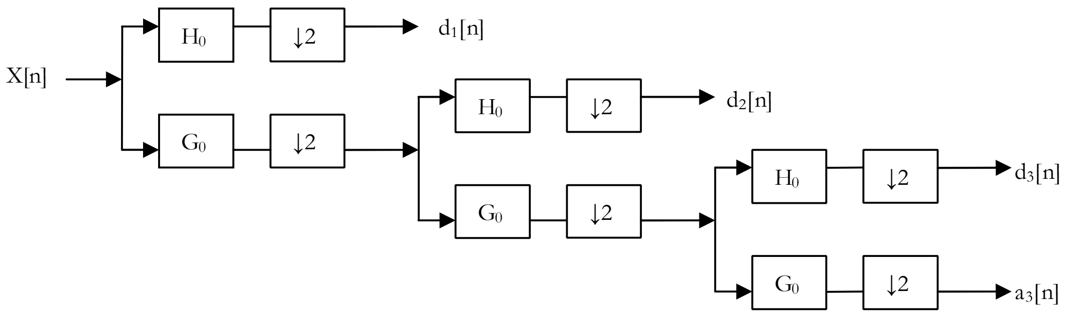


Figure 3 Three-level wavelet decomposition tree.

At each decomposition level, the half band filters produce signals spanning only half the frequency band. This doubles the frequency resolution as the uncertainity in frequency is reduced by half. In accordance with Nyquist's rule if the original signal has a highest frequency of ω, which requires a sampling frequency of 2ω radians, then it now has a highest frequency of ω/2 radians. It can now be sampled at a frequency of ω radians thus discarding half the samples with no loss of information. This decimation by 2 halves the time resolution as the entire signal is now represented by only half the number of samples. Thus, while the half band low pass filtering removes half of the frequencies and thus halves the resolution, the decimation by 2 doubles the scale.

With this approach, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. The time-frequency plane is thus resolved as shown in figure 1(d) of Chapter 1. The filtering and decimation process is continued until the desired level is reached. The maximum number of levels depends on the length of the signal. The DWT of the original signal is then obtained by concatenating all the coefficients, a[n] and d[n], starting from the last level of decomposition.

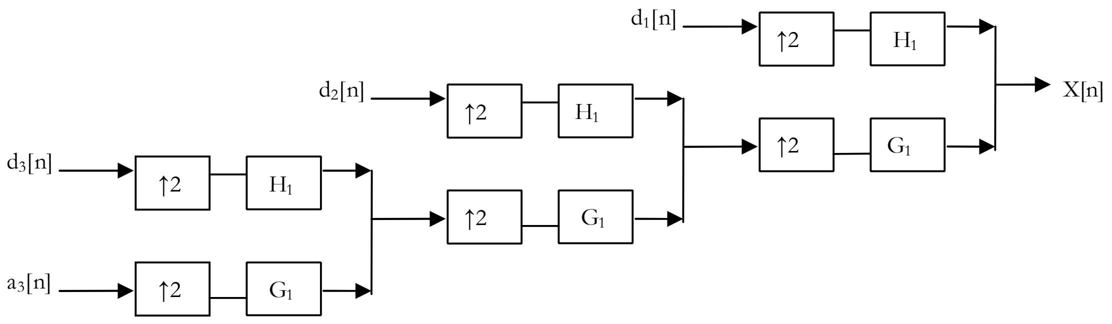


Figure 4 Three-level wavelet reconstruction tree.

Figure 4 shows the reconstruction of the original signal from the wavelet coefficients. Basically, the reconstruction is the reverse process of decomposition. The approximation and detail coefficients at every level are upsampled by two, passed through the low pass and high pass synthesis filters and then added. This process is continued through the same number of levels as in the decomposition process to obtain the original signal. The Mallat algorithm works equally well if the analysis filters, G_0 and H_0, are exchanged with the synthesis filters, G_1 and H_1.

2.4.2 Classification Of Wavelets

We can classify wavelets into two classes: (a) orthogonal and (b) biorthogonal. Based on the application, either of them can be used.

2.4.2.1 Features of orthogonal wavelet filter banks

The coefficients of orthogonal filters are real numbers. The filters are of the same length and are not symmetric. The low pass filter, G_0 and the high pass filter, H_0 are related to each other by

$$H_0 (z) = z^{-N} G_0 (-z^{-1}) \qquad 2.4$$

The two filters are alternated flip of each other. The alternating flip automatically gives double-shift orthogonality between the lowpass and highpass filters [1], i.e., the scalar product of the filters, for a shift by two is zero. i.e., $\sum G[k]\, H[k-2l] = 0$, where $k,l \in Z$ [4]. Filters that satisfy equation 2.4 are known as Conjugate Mirror Filters (CMF).
Perfect reconstruction is possible with alternating flip.

Also, for perfect reconstruction, the synthesis filters are identical to the analysis filters except for a time reversal. Orthogonal filters offer a high number of vanishing moments. This property is useful in many signal and image processing applications. They have regular structure which leads to easy implementation and scalable architecture.

2.4.2.2 Features of biorthogonal wavelet filter banks

In the case of the biorthogonal wavelet filters, the low pass and the high pass filters do not have the same length. The low pass filter is always symmetric, while the high pass filter could be either symmetric or anti-symmetric. The coefficients of the filters are either real numbers or integers.

For perfect reconstruction, biorthogonal filter bank has all odd length or all even length filters. The two analysis filters can be symmetric with odd length or one symmetric and the other antisymmetric with even length. Also, the two sets of analysis and synthesis filters must be dual. The linear phase biorthogonal filters are the most popular filters for data compression applications.

2.4.3 Wavelet Families

There are a number of basis functions that can be used as the mother wavelet for Wavelet Transformation. Since the mother wavelet produces all wavelet functions used in the transformation through translation and scaling, it determines the characteristics of the resulting Wavelet Transform. Therefore, the details of the particular application should be taken into account and the appropriate mother wavelet should be chosen in order to use the Wavelet Transform effectively.

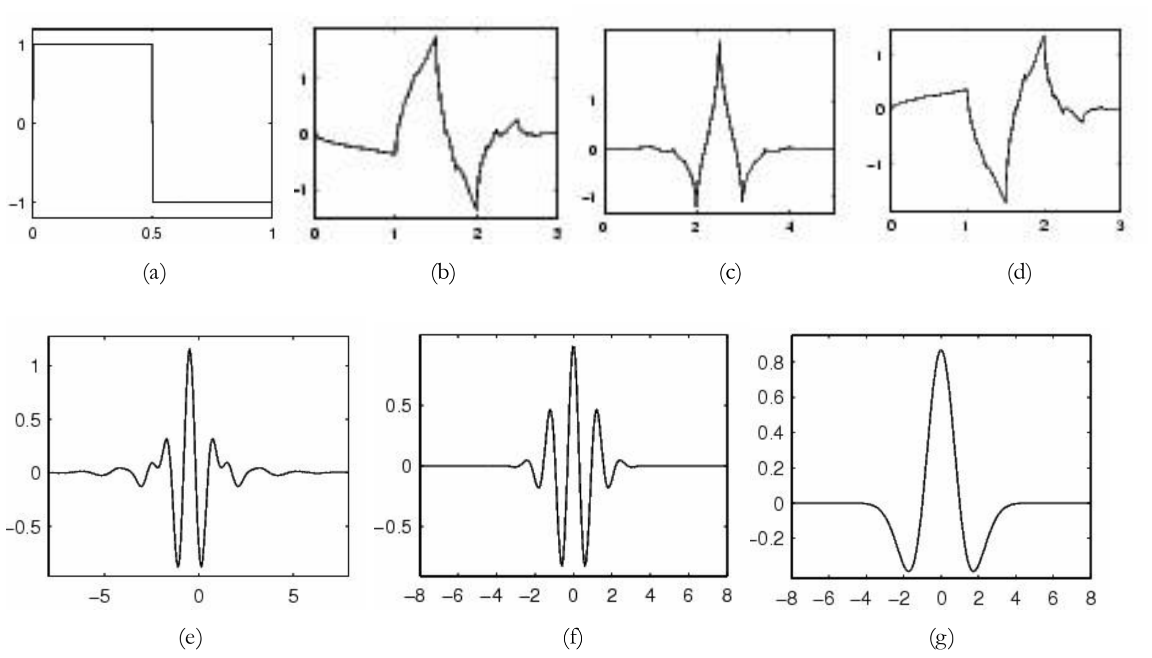


Figure 5 Wavelet families (a) Haar (b) Daubechies4 (c) Coiflet1 (d) Symlet2 (e) Meyer (f) Morlet (g) Mexican Hat.

Figure 5 illustrates some of the commonly used wavelet functions. Haar wavelet is one of the oldest and simplest wavelet. Therefore, any discussion of wavelets starts with the Haar wavelet. Daubechies wavelets are the most popular wavelets. They represent the foundations of wavelet signal processing and are used in numerous applications. These are also called Maxflat wavelets as their frequency responses have maximum flatness at frequencies 0 and π. This is a very desirable property in some applications. The Haar, Daubechies, Symlets and Coiflets are compactly supported orthogonal wavelets. These wavelets along with Meyer wavelets are capable of perfect reconstruction. The Meyer, Morlet and Mexican Hat wavelets are symmetric in shape. The wavelets are chosen based on their shape and their ability to analyze the signal in a particular application.

2.5. Applications

There is a wide range of applications for Wavelet Transforms. They are applied in different fields ranging from signal processing to biometrics, and the list is still growing. One of the prominent applications is in the FBI fingerprint compression standard. Wavelet Transforms are used to compress the fingerprint pictures for storage in their data bank. The previously chosen Discrete Cosine Transform (DCT) did not perform well at high compression ratios. It produced severe blocking effects which made it impossible to follow the ridge lines in the fingerprints after reconstruction. This did not happen with Wavelet Transform due to its property of retaining the details present in the data.

In DWT, the most prominent information in the signal appears in high amplitudes and the less prominent information appears in very low amplitudes. Data compression can be achieved by discarding these low amplitudes. The wavelet transforms enables high compression ratios with good quality of reconstruction. At present, the application of wavelets for image compression is one the hottest areas of research. Recently, the Wavelet Transforms have been chosen for the JPEG 2000 compression standard.

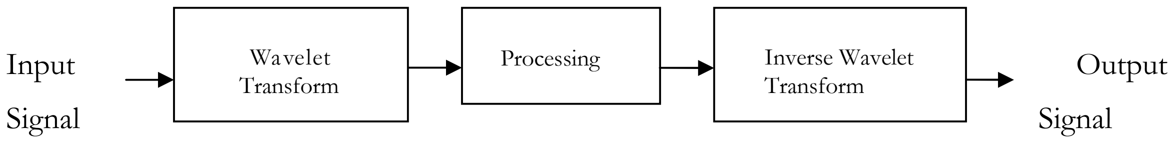


Figure 6 Signal processing application using Wavelet Transform.

Figure 6 shows the general steps followed in a signal processing application. Processing may involve compression, encoding, denoising etc. The processed signal is either stored or transmitted. For most compression applications, processing involves quantization and entropy coding to yield a compressed image. During this process, all the wavelet coefficients that are below a chosen threshold are discarded. These discarded coefficients are replaced with zeros during reconstruction at the other end. To reconstruct the signal, the entropy coding is decoded, then quantized and then finally Inverse Wavelet Transformed.

Wavelets also find application in speech compression, which reduces transmission time in mobile applications. They are used in denoising, edge detection, feature extraction, speech recognition, echo cancellation and others. They are very promising for real time audio and video compression applications. Wavelets also have numerous applications in digital communications. Orthogonal Frequency Division Multiplexing (OFDM) is one of them. Wavelets are used in biomedical imaging. The popularity of Wavelet Transform is growing because of its ability to reduce distortion in the reconstructed signal while retaining all the significant features present in the signal.

CHAPTER.3

2.5.1.1 Systolic Array

2.6. Introduction

Systolic arrays probably constitute a perfect kind of special purpose computer. In their simplest appearance, they may provide only one operation that is repeated over and over again. Yet, systolic arrays show an abundance of practice-oriented applications, mainly in fields dominated by iterative procedures: numerical mathematics, combinatorial optimisation, linear algebra, algorithmic graph theory, image and signal processing, speech and text processing, et cetera.

For a systolic array can be tailored to the structure of its one and only algorithm accurately. So that time and place of each executed operation are fixed once and for all. And communicating cells are permanently and directly connected, no switching required. The algorithm has in fact become hardwired. Systolic algorithms in this respect are considered to be **hardware algorithms**.

Systolic architecture consists of an array of processing elements, where data flows between neighboring elements, synchronously, from different directions. Processing element takes data from Top, Left and output the results to Right, Bottom.

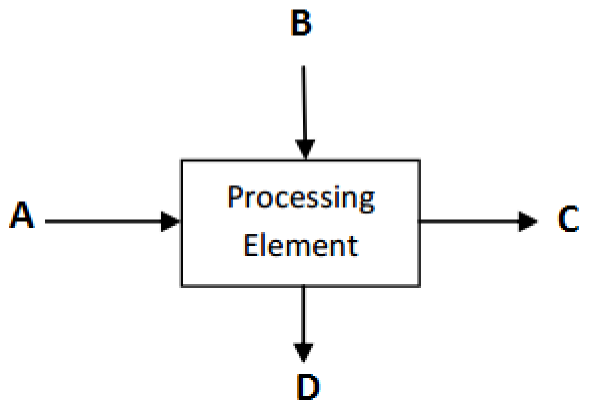


Figure 7 Systolic Array PE block

One of the key application of Systolic architecture is matrix multiplication. Here each processing element performs four operations, namely **FETCH, MULTIPLICATION, SHIFT & ADDITION.** As following figure depicts, "in_a", "in_b" are inputs to the processing element and "out_a", "out_b" are outputs to the processing element. "out_c" is to get the output result of each processing element.

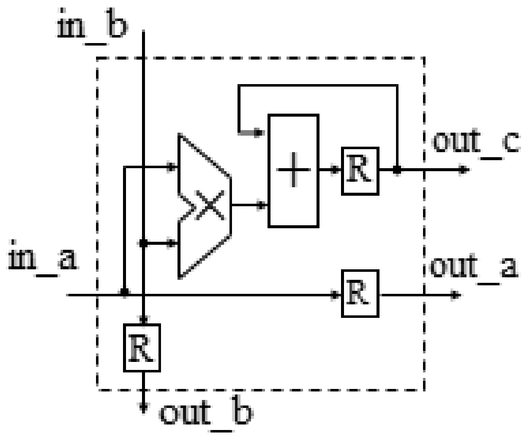


Figure 8 Systolic Array Architecture.

Processing elements are arranged in the form of an array. In this case we analyze, multiplication of 3x3 matrices, which can be easily extended. Let say the two matrices are A and B. Following figure depicts how matrix A and B are fed into PE(processing element) array.

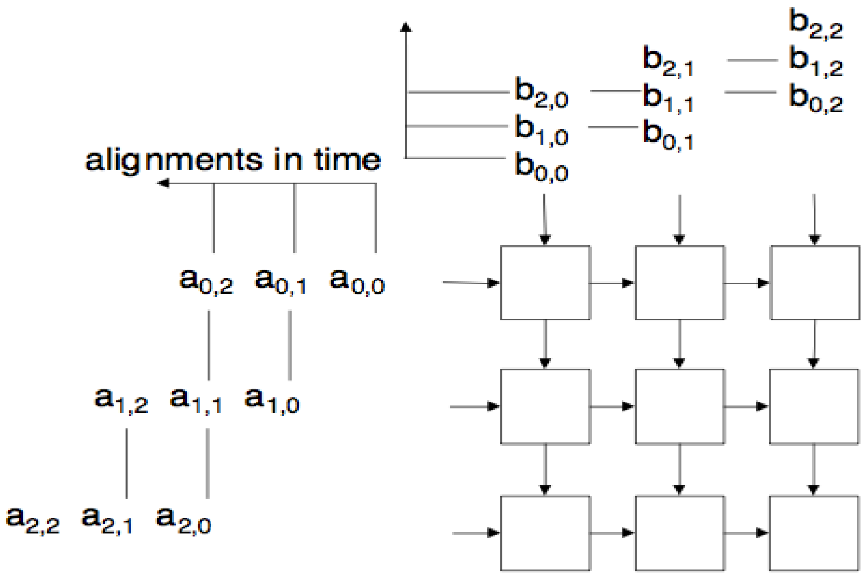

$b_{2,2}$
$b_{2,1}$ $b_{1,2}$
$b_{2,0}$ $b_{1,1}$ $b_{0,2}$
$b_{1,0}$ $b_{0,1}$
$b_{0,0}$
alignments in time
$a_{0,2}$ $a_{0,1}$ $a_{0,0}$
$a_{1,2}$ $a_{1,1}$ $a_{1,0}$
$a_{2,2}$ $a_{2,1}$ $a_{2,0}$

Figure 9 Systolic Array Functioning

CHAPTER.4

3. LITERATURE SURVEY

Memory requirements for storing intermediate signals and critical path are essential issues for 2-D DWT [1] .chin proposed 2-D dual-mode LDWT architecture has the merits of low transpose memory (TM), low latency, and regular signal flow, making it suitable for very large-scale integration implementation .In DWT implementation, a 1-D DWT needs massive computation thus, the computation unit dominates most of the hardware cost. The computation and the access of the memory take time, and therefore affect the latency that is long IRSA is employed to reduce the required Transpose memory.

A novel parallel stripe-based scanning method based on the analysis of the dependency graph of the lifting scheme is proposed by Yusong Hu [2]. The elimination of frame memory and the small temporal memory lead to significant reduction in overall size and have a regular structure and achieved 100% hardware utilization. The overlapped stripebased scanning method we used in the proposed design has scalable stripe width with 7 columns overlapped between two adjacent stripes. The data in the overlapped columns are processed by a newly proposed partial processor in the rDWT of the first level (Level 1) DWT. With this approach, the temporal memory of Level 1 DWT, which occupies a dominating chip area in other lifting-based architectures, is eliminated at the expenses of reading seven more pixels per cycle and a few extra arithmetic resources. The elimination of the temporal memory results in significant memory saving. Pipelined architecture, which does not require frame memories, unlike the existing folded multi-level DWT architectures and the flipping method is applied in their design for shortening the critical path delay.

Basant Kumar Mohanty [3] proposed a design strategy for the derivation of memory efficient architecture for multilevel 2-D DWT he proposed design scheme on a convolutionbased generic architecture for the computation of three-level 2-D DWT based on Daubechies The proposed structure does not involve frame buffer

An hardware efficient parallel fir structure with parallel structure is implemented by k k parhi [4] he achieved a high speed and low computation time and higher processing speed can be achieved by using parallel fir filter structures but hardware cost increases. The proposeddesign can also save a large amount of multipliers and storage elements**.** The throughput rate is improved by a factor of 4 by the proposed design, but the hardware cost increases by a factor of around 3.

A novel architecture for DWT that can be reconfigured to be adapted to different sizes of input .the architecture can be reconfigured to 3 modes 1-D and 2-D DWT is proposed by Qin sung [5].The reconfigurable architecture mainly on convolution based approach, which has better scalability. In order to minimize the critical path multipliers and adders are pipelined and data dependencies can be overcome by loop unrolling technique

An efficient architecture for the two-dimensional discrete wavelet transform 2-D DWT is proposed by Po-Cheng Wu [6]. The architecture includes a transform module, a RAM module, and a multiplexer. In the first-level decomposition, the multiplexer selects data from the input image. The transform module decomposes the input image to the four sub bands. The advantage of such a scheme is that the data flow is very regular. the transform module is tree-structured and comprises two stages. Stage 1 performs horizontal filtering, and stage 2 performs vertical filtering.

An efficient multi-input-multi-output VLSI architecture (MIMOA) for twodimensional lifting-based discrete wavelet transform is proposed by Xin Tian[7]. Computing time of MIMOA is reduced with less increase of hardware cost MIMOA has the least consumption of hardware cost and on-chip memory. However, the best advantage of this architecture is that it provides a variety of hardware implementations to meet different processing speed requirements by selecting different throughput rates.

A novel 2-D DWT architecture are composed of two 1-D DWT cores and a 2 × 2 transposing register array is proposed by Yeong-Kang Lai [8] . In this architecture 1-D DWT core consumes two input data and produces two output coefficients per cycle, and its critical path takes one multiplier delay only. Two coefficients at the same column are scanned along the row direction and fed into the column processor with the proposed parallel scanning method.

Chao-Tsung Huang [9] flipping structure, is proposed for the lifting-based discrete wavelet transform. It can provide a variety of hardware implementations to improve and possibly minimize the critical path as well as the memory requirement of the lifting- based discrete wavelet transform by flipping conventional lifting structures. Since the timing problem is due to the accumulation of timing delays from the input node to the computation node in each computing unit, releasing the accumulation by eliminating the multipliers on the path from the input node to the computation node. This can be achieved by flipping each computing unit with the inverse of the multiplier coefficient Moreover, the computation nodes can be split into two parts: One is the summation of the multiplication results from register nodes and the other one is the adder on the accumulative path. The timing accumulation can be greatly reduced by flipping the original lifting-based architectures. An- other advantage of flipping structures is that no additional multipliers will be required if the computing units are all flipped

Francescomaria Marino [10] proposed two scalable architectures that perform the discrete wavelet transform DWT of an N sample sequence in only N/2 clock cycles. This result has been achieved by means of a carefully balanced pipelining, First, Architecture 1 and Architecture 2 can be employed for performing two times faster processing than allowed by other architectures working at the same clock frequency (high- speed utilization). Second, they can be employed even using a two times lower clock frequency but reaching the same performance as other architectures. This second possibility allows for reducing the supply voltage and the power dissipation, respectively, by a factor of two and four with respect to other architectures (low-power utilization).

Reconfigurable Array Targeting Discrete Wavelet Transform for System-on-Chip Applications is proposed by Georgi [11].Reconfigurable architectures are highly suitable for complex algorithms which are part of changing standards like JPEG2000 .reconfigurable arrays are for one particular domain of applications, which provide high performance over generic Field Programmable Gate Arrays (FPGAs) he proposed reconfigurable array is flexible to implement lifting and integer based different DWT algorithm.

3.1. Summary Of Literature Survey

A detailed literature survey has done on the architecture of 1-D and 2-D DWT .for designing and implementing DWT on VLSI architectures is to reduce on chip memory usage and computations should be lower. A parallel implementation can give higher speed and increase cost an efficient design in parallel fir structure can reduce area and cost. Based on the data scanning methods can reduce the memory on chip and power efficiency can be achieved. By using flipping architectures we can reduce the critical path of the filters.

Architecture	Yusung	Chinhsien	K parhi	Basant	Jiang
Category	Lifting	Lifting	Lifting	Convolution	Convolution
Data Scanning	Strip Based	Line Based	Line Based	Line Based	Line Based
Frame Memory	Yes	Yes	Yes	No	Yes
Temporal Memory	No	Yes	Yes	Yes	Yes
Flipping	Yes	No	NO	Yes	-
Parallel Architecture	Yes	No	Yes	Yes	-

Table 1 Comparison Of Architectures

From the literature survey mainly 2 architectures are used for implementation of DWT i.e. convolution based and lifting scheme. While the convolution based architectures are implemented with FIR filter banks, the lifting-based architectures are implemented by factorizing the filter banks into several lifting steps followed by a scaling step. Both types of architectures are composed of arithmetic resources such as multipliers, adders and multiplexers, and storage resources. The storage resources include transposition memory, temporal memory and frame memory.

CHAPTER.5

4. SYSTOLIC ARRAY ARCHITECTURE FOR DWT

The design of DWT-SA is based on computation schedule derived from Equations 2a-2n, which are the result of applying the pyramid algorithm for N=8 to the six stage filter. As shown in Equations 2a-2n there are eight first octave coefficients computations, four second octave coefficients and two third octave computations The DWT-SA architecture comprises of the following basic blocks:

4.1. The Filter Unit (Fu)

Equations 1a-1b show that the high pass and low pass filter coefficients can be computed using a six stage multiply and accumulate digital filter, where partial results are computed at each stage separately and then progressively passed to the next higher stage for accumulation. Equations 1a-1b suggests that due to filter coefficient calculations spanning over 5 clock cycles, a temporary storage is required for the incoming data samples. The systolic architecture of a six-stage filter is shown in Figure 10. The latency of each filter stage is 1.

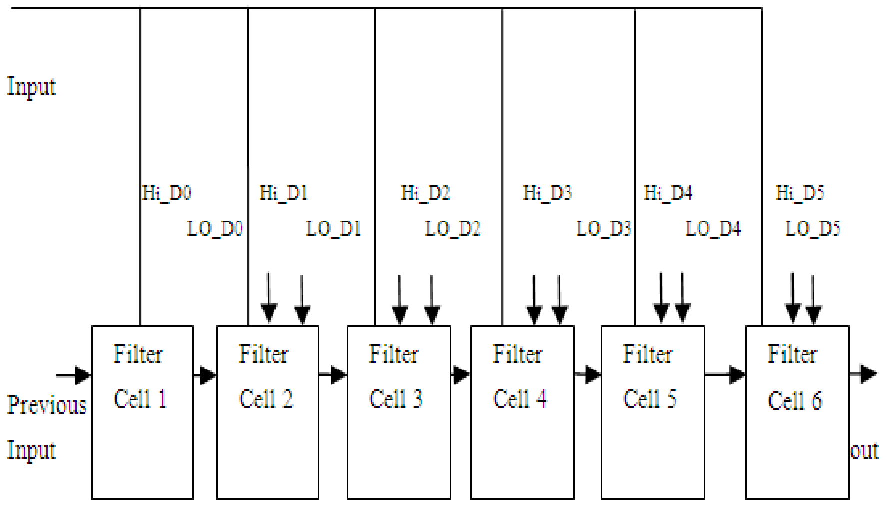


Figure 10 Six-stage non-recursive fir digital filter

The filter behaves in a systolic manner by computing partial products of more than one coefficient at a time. The first multiply and accumulate stage computes the first partial product and passes it to the second filter stage where it is added to the second partial product. That repetitive action continues until a complete DWT coefficient is out from the sixth filter stage.

The filter behaves in a systolic manner by computing partial products of more than one coefficient at a time. The first multiply and accumulate stage computes the first partial product and passes it to the second filter stage where it is added to the second partial product. That repetitive action continues until a complete DWT coefficient is out from the sixth filter stage.

The filter cell consist of only one multiplier, one adder and two registers to store high pass and low pass coefficients. In filter cell, a signed number multiplication problem occurs. The signed-number represents either positive, negative numbers or one positive and other negative numbers. To avoid this problem, the proposed filter cell consists of invert and xor operation as shown in Figure 11

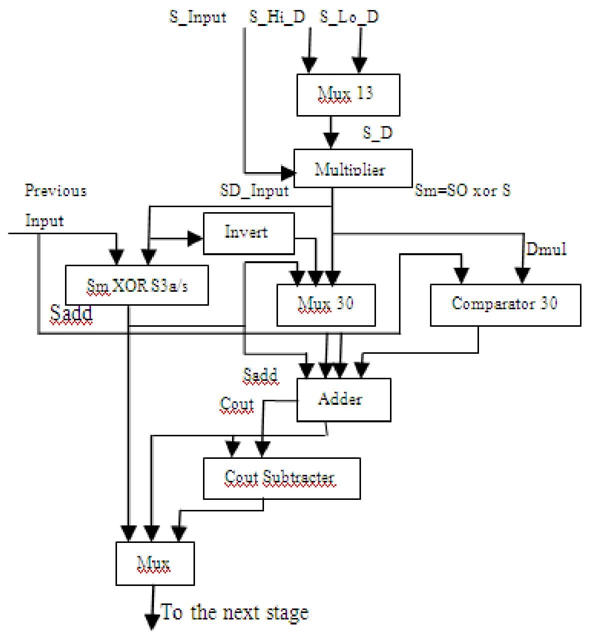


Figure 11 Systolic Array Architecture

4.2. <u>The Storage Unit</u>

Two storage units are used in the proposed architecture.

• Input Delay Unit (ID) and

• Register Bank (RB)

4.2.1.1.1 Input Delay Unit (ID):

The data registers and input delay used in theses storage units have been constructed from standard D-latch. Equations 2a-2n shows that the value of computed filter coefficient depends on the present as well as the five previous data samples. The negative time indexes in Equations 2 correspond to the reference starting time unit 0. Figure 12 shows the block diagram of the input delay unit

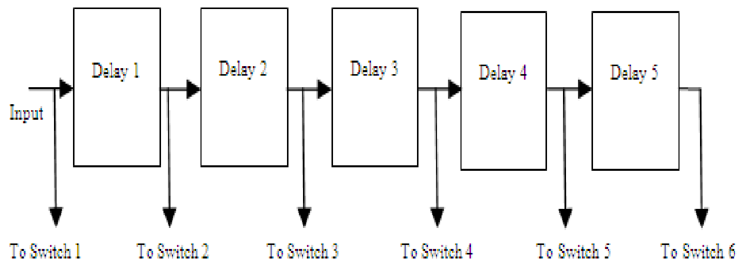


Figure 12 Input Delay unit

Input Delay Unit

As shown in Figure 12 five delays are connected serially. At any clock cycle each delay passes its contents to its right neighbor which results in only five past values being retained. The input of delay is applied to the switch.

4.2.1.1.2 Register Bank (RB)

In proposed architecture number of registers required is 26.Here, two 13 registers architecture are connected serially and constitute one register bank and are controlled by an overall clock. The output of one register is directly connected to the input of the adjacent register, and thus the register bank is implemented as shown in Figure 13.

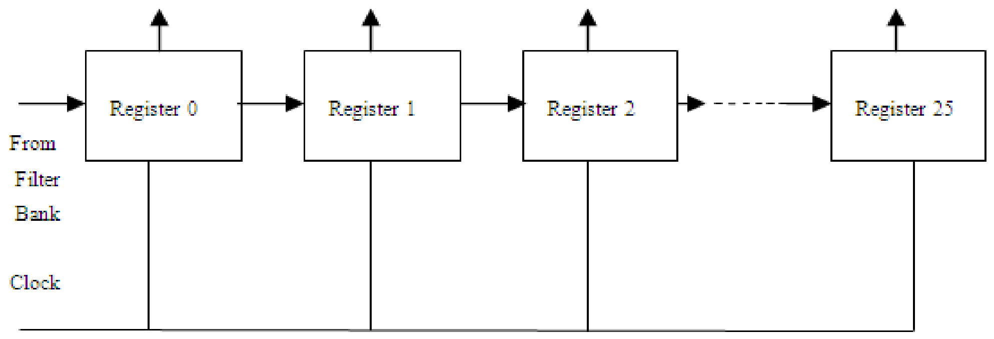


Figure 13 Register Bank Unit

4.3. Control Unit

In DWT-SA architecture, computations are scheduled at the earliest possible clock cycle, and computed output samples are available one clock cycle after they have been scheduled as shown in TABLE I. The delay is minimized through the pipeline facilitating real time operation. The computation schedule in TABLE 2 corresponds to a high hardware utilization of more than 85%. All the intermediate and the associated periods of activity are listed in TABLE 3. The complete design of the control unit for DWT-SA is shown in Figure 14.

Cycle	High-Pass	Low-Pass
1	b(0)	c(0)
2	--	--
3	b(2)	c(2)
4	d(0)	e(0)
5	b(4)	c(4)
6	--	--
7	b(6)	c(6)
8	d(4)	e(4)
9	b(0)	c(0)

Table 2 Schedule for one complete set of computations.

Samples	Available Cycle	Life Period
c(0)	1	1 to 12
c(2)	3	3 to 14
c(4)	5	5 to 16
c(6)	7	7 to 18
e(0)	12	12 to 8
e(4)	16	16 to 8

Table 3 Activity periods for intermediate results

The control unit uses switch, decoder, and four FSM. The switching action is done by using FSM. CU directs data from the Input Delay or the register bank to the filter unit. The CU multiplexes data from the ID every second cycle, and from the RB in cycles 4, 6, and 8.In cycle2, 6, CU remains idle, i.e.it does not allow any passage of data. Proper timing synchronization as well as enabling and disabling of the CU are insured by the CLK signal.

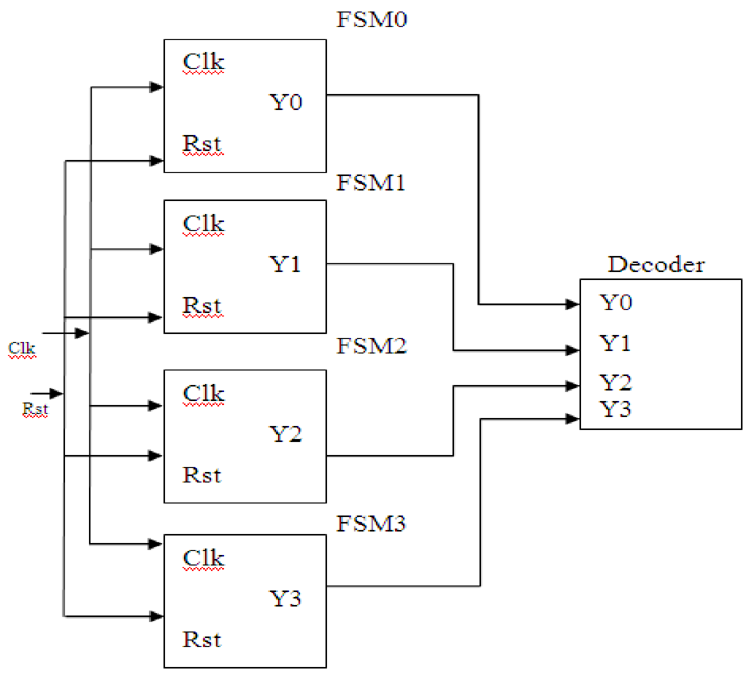


Figure 14 Control unit for DWT-SA

Control logic consists of four FSM. The switch is operated on the state diagram of FSM. According to that it accepts data from input delay and register bank. Figure 15 shows the state diagram for FSM0.Here, there are two states, S1 and S2.When reset is 1, it produces output 1.Otherwise it produces 0

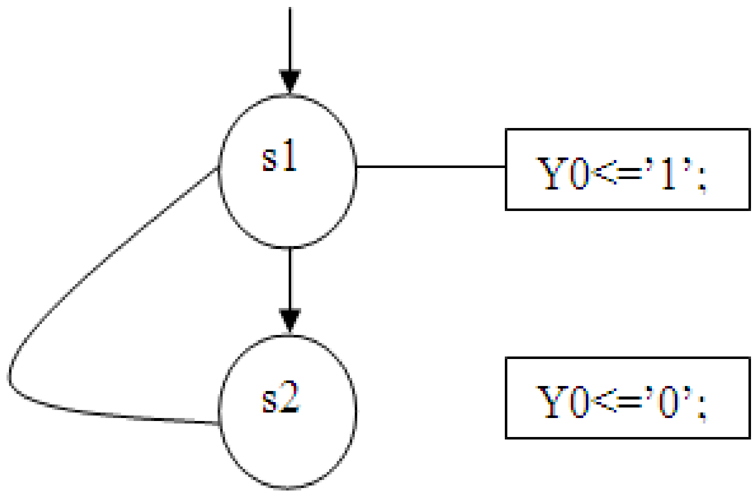


Figure 15 The State Diagram For FSM0.

Figure 16 shows the state diagram for FSM1. It consists of four states S1, S2, S3, S4.When reset is 1, and the output is 0 for first clock. After that it goes to the next state, i.e. S2.At that time clock is incremented by 1.When clock cycle is less than 2,it produces output 0.But,when clock is greater than 2 then it goes to the next state. In S3 state, it produces output 1.This process is repeated for clock 3, 4, 5, 6, and 7. If clock is 978-1-4799-7678-2/15/$31.00 ©2015 IEEE greater than 7 then it again comes to the S3 state and produces the output.

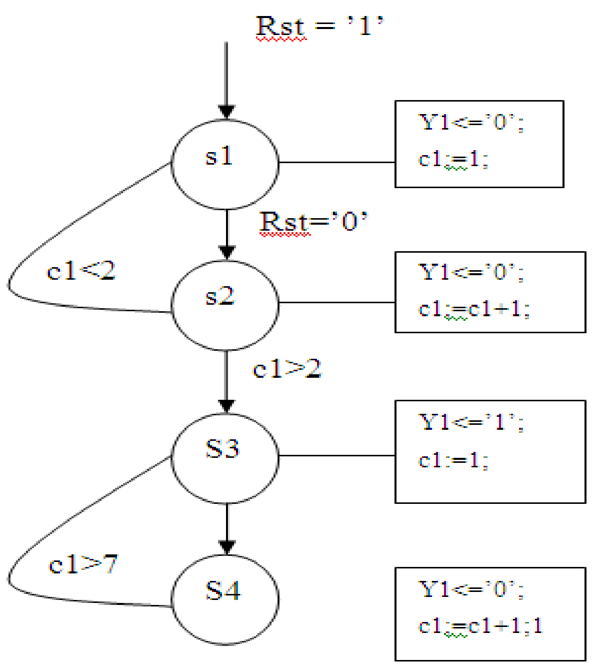


Figure 16 The State Diagram For FSM1

Figure 17 shows state diagram for FSM2.When reset is 0,state S2 produces output 0.This occurs when clock is less than 6.For the next clock i.e. Clock is greater than 6,the state S3 produce output 1.But when clock is greater than 7,then sate S4 changes its state to the previous state S3.

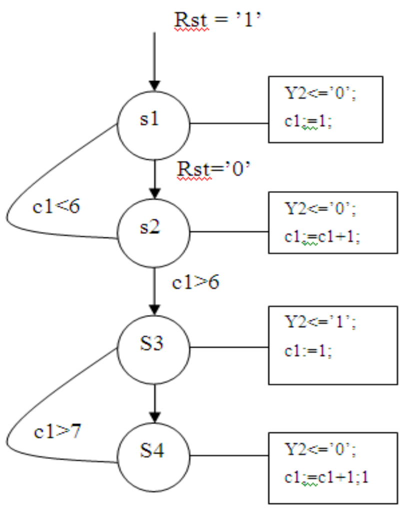


Figure 17 The State Diagram For FSM2.

The next is FSM3, which consist of S1, S2, S3, and S4. Figure 18 shows the sate diagram for FSM3.When reset is 1, output y3 produces 0.In sate S2, when clock cycle is less than 8, then output is 0.But in the next clock cycle the output is high. If clock cycle is greater than 7, it changes the state S4 to S3 and produces output high. In all this FSM, outputs are connected to decoder. The output of decoder is connected to the select line of switch, by selecting particular select line; switching action of switch is done. And according to that switch, the data is applied to the filter cell.

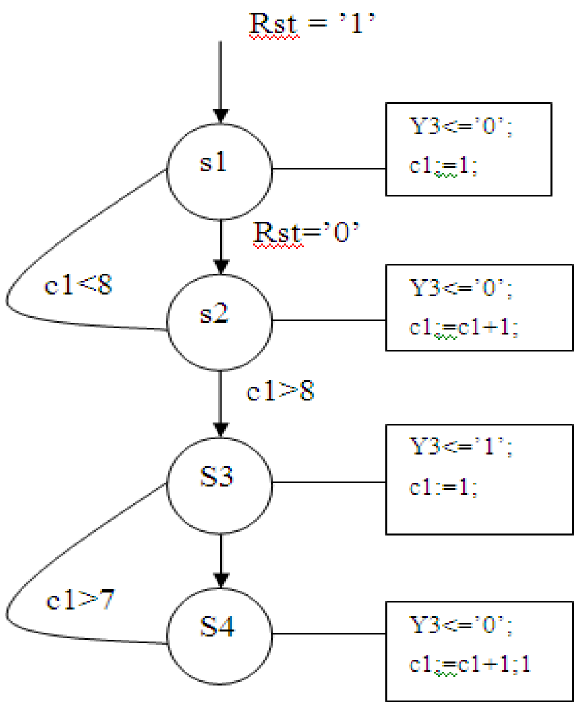


Figure 18 The State Diagram For FSM3.

4.4. Register Allocation

In the DWT-SA architecture Control Unit (CU) and the Register Bank (RB) synchronize the availability of operands. There are two schemes which can be employed for this purpose, namely

• Forward Register Allocation (FRA)

• The Forward-Backward Register Allocation(FBRA)

The FRA method uses a set of registers which are allocated to intermediate data on the first come first serve basis. It does not reassign any registers to other operands once its contents have been accessed. The FBRA scheme is similar, except that once the operand stored has been used; there register is allocated to another operand. The FRA method is simpler, requires less control circuitry and permits easy adaptation of this architecture for coefficient calculation of more than 3 octaves. It results however in less efficient register utilization.

4.5. Dwt-Sa Architecture

The proposed architecture is shown in Figure 5.12. It is obtained by systematic analysis of the FRA register allocation in conjunction with the filter equations 1a and 1b .Delay of the DWT-SA architecture consists of the latency period necessary to fill up the filter for the first time, plus the delay through the registers as described in Table 1. The output coefficients are obtained from the final filter stage.

The architecture is optimized in hardware by using only a single multiplier and adder set in each filter cell to generate all high pass and low pass coefficient. The coefficients which are present at the output of filter are stored in register.The control logic controls the switching action of switch. The data which are coming from delay input and register bank is controlled by switch. According to the position of switch, one of the data is select and performs filtering operations. This process is continued up to the 53 clock cycle.

To select the particular output coming from filter unit, one of the small unit is used. When select line is zero, it selects one data. Force zero blocks is used to set the values zero. For controlling of this unit FSM is used. The select line is connected to the FSM. When reset is zero, it produces output. Otherwise it produces 0.This results in a simple and efficient systolic implementation for the computation of 1D-DWT and hence is suitable for VLSI implementation.

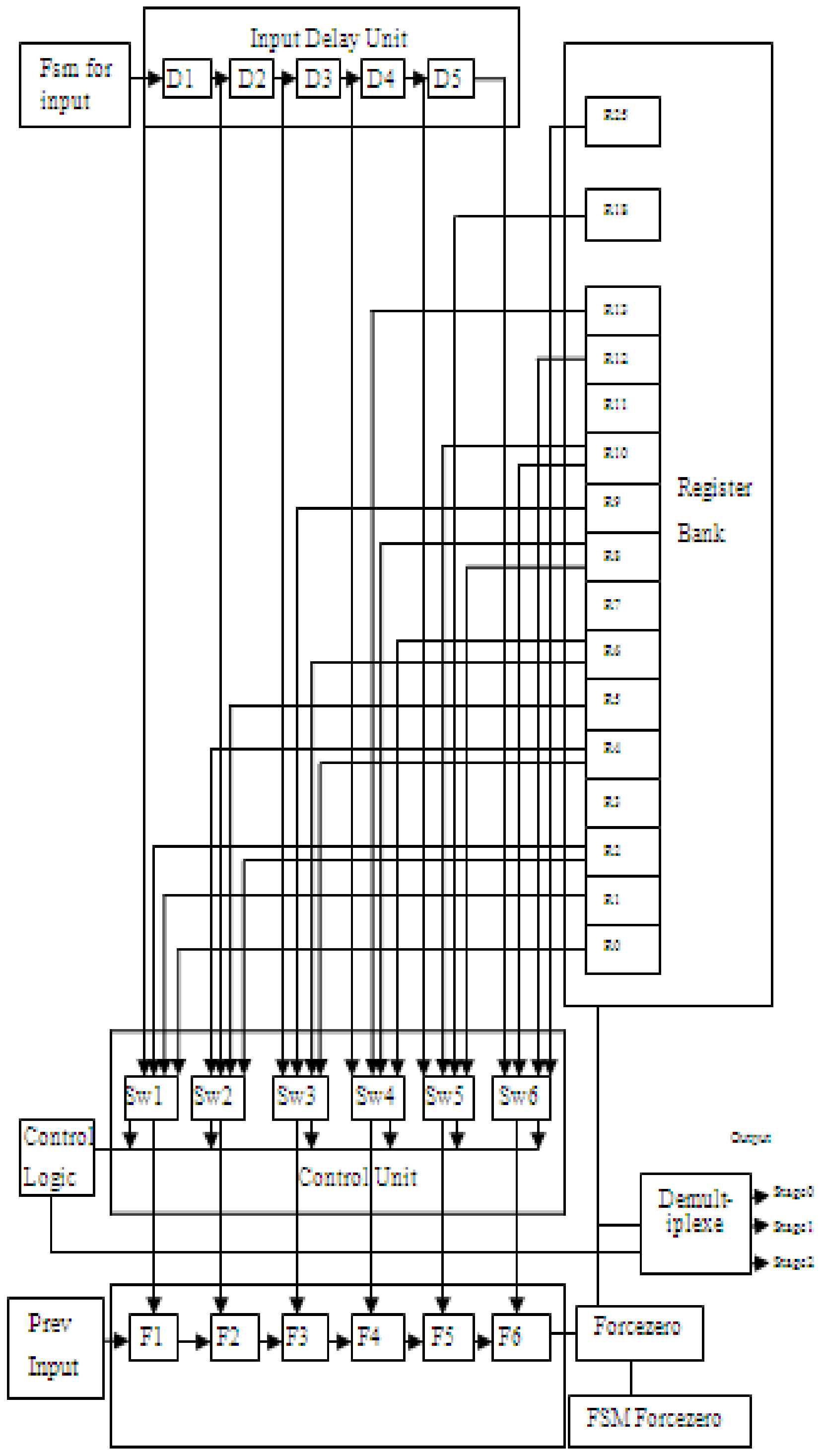


Figure 19 The Proposed Dwt-SA Architecture IEEE

4.6. General Implementation Flow:

The generalized implementation flow diagram of the project is represented as:

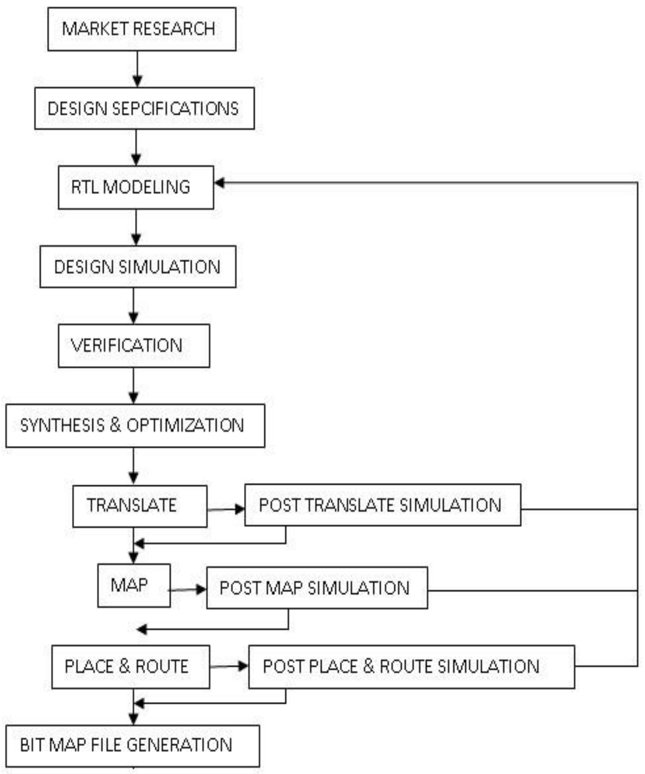


Figure 20 General Implementation Flow Diagram

Initially the market research should be carried out which covers the previous version of the design and the current requirements on the design. Based on th i s survey, the specification and the architecture must be identified. Then the RTL modeling should be carried out in VERILOG HDL with respect to the identified architecture. Once the RTL modeling is done, it should be simulated and verified for all the cases. The functional verification should meet the intended architecture and should pass all the test cases.

Once the functional verification is clear, the RTL model will be taken to the synthesis process. Three operations will be carried out in the synthesis process such as

- Translate
- Map
- Place and Route

The developed RTL model will be translated to the mathematical equation format which will be in the understandable format of the tool. These translated equations will be then mapped to the library that is, mapped to the hardware. Once the mapping is done, the gates were placed and routed. Before these processes, the constraints can be given in order to optimize the design. Finally the BIT MAP file will be generated that has the design information in the binary format which will be dumped in the FPGA board

CHAPTER.6

5. IMPLEMENTATION RESULTS AND DISCUSSIONS

The DWT process and the developed architecture for the required functionality were discussed in the previous chapters. Now this chapter deals with the simulation and synthesis results of the DWT process. Here Isim tool is used in order to simulate the design and checks the functionality of the design. Once the functional verification is done, the design will be taken to the Xilinx tool for Synthesis process and the netlist generation.

The Appropriate test cases have been identified in order to test this modelled DWT process architecture. Based on the identified values, the simulation results which describes the operation of the process has been achieved. This proves that the modelled design works properly as per its functionality.

5.1. Simulation Results

The test bench is developed in order to test the modeled design. This developed test bench will automatically force the inputs and will make the operations of algorithm to perform. **Behavioral Simulation** (RTL Simulation):

This is first of all simulation steps; those are encountered throughout the hierarchy of the design flow. This simulation is performed before synthesis process to verify RTL (behavioral) code and to confirm that the design is functioning as intended. Behavioral simulation can be performed on either VHDL or Verilog designs. In this process, signals and variables are observed, procedures and functions are traced and breakpoints are set. This is a very fast simulation and so allows the designer to change the HDL code if the required functionality is not met with in a short time period. Since the design is not yet synthesized to gate level, timing and resource usage properties are still unknown.

5.2. DWT Block

The initial block of the design is that the Discrete Wavelet Transform (DWT) block which is mainly used for the transformation of the image. In this process, the image will be transformed and hence the high pass coefficients and the low pass coefficients were generated. Since the operation of this DWT block has been discussed in the previous chapter, here the snapshots of the simulation results were directly taken in to consideration and discussed.

When ever the input is send, the data divided into even data and odd data. The even data and odd data is stored in the temporary registers. When the reset is high the temporary register value consists of zero when ever the reset is low the input data split into the even data and odd data. The input data read up to sixteen clock cycles after that the data read according to the lifting scheme. The output data consists of low pass and high pass elements. This is the 1-D discrete wavelet transform. The 2-D discrete wavelet transform is that the low pass and the high pass again divided into LL, LH and HH, HL. The output is verified in the ISim.

For this DWT block, the clock and reset were the primary inputs. The pixel values of the image, that is, the input data will be given to this block and hence these values will be split in to even and odd pixel values. In the design, this even and odd were taken as a array which will store its pixel values in it and once all the input pixel values over, then load will be made high which represents that the system is ready for the further process.

Once the load signal is set to high, then the each value from the even and odd array will be taken and used for the Low Pass Coefficients generation process. Hence each value will be given to the adder and in turn given to the multiplication process with the filter coefficients. Finally the Low Pass Coefficients will be achieved from the addition process of multiplied output and the odd pixel value.

Again this Low Pass Coefficient will be taken and it will be multiplied with the filter coefficients. The resultant will be added with the even pixel value which gives the High Pass Coefficient. Hence all the values from even and odd array will be taken and then above said process will be carried out in order to achieve the High and Low Pass Coefficients of the image.

Now these low pass coefficients and the high pass coefficients were taken as the input for the further process. Hence for the DWT-2 process, low pass coefficients will be taken as the inputs and will do the process in order to calculate the low pass and high pass coefficients from the transformed coefficients of DWT-1. In DWT-2, the same process as in DWT-1 will be carried out. Hence the simulated waveform is shown in the figure 6.5

5.3. Introduction To FPGA

FPGA contains a two dimensional arrays of logic blocks and interconnections between logic blocks. Both the logic blocks and interconnects are programmable. Logic blocks are programmed to implement a desired function and the interconnects are programmed using the switch boxes to connect the logic block. To be more clear, if we want to implement a complex design (CPU for instance), then the design is divided into small sub functions and each sub function is implemented using one logic block. Now, to get our desired design (CPU), all the sub functions implemented in logic blocks must be.

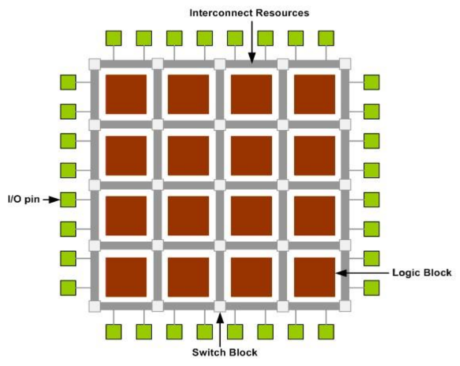


Figure 21 Internal Architecture of FPGA

FPGAs, alternative to the custom ICs, can be used to implement an entire System On one Chip (SOC). The main advantage of FPGA is ability to reprogram. User can reprogram an FPGA to implement a design and this is done after the FPGA is manufactured. This brings the name "Field Programmable." Custom ICs are expensive and takes long time to design so they are useful when produced in bulk amounts. But FPGAs are easy to implement with in a short time with the help of Computer Aided Designing (CAD) tools (because there is no physical layout process, no mask making, and no IC manufacturing).

Some disadvantages of FPGAs are, they are slow compared to custom ICs as they can't handle vary complex designs and also they draw more power.

Xilinx logic block consists of one Look Up Table (LUT) and one FlipFlop. An LUT is used to implement number of different functionality. The input lines to the logic block go into the LUT and enable it. The output of the LUT gives the result of the logic function that it implements and the output of logic block is registered or unregistered out put from the LUT.

SRAM is used to implement a LUT.A k-input logic function is implemented using 2^k * 1 size SRAM. Number of different possible functions for k input LUT is 2^2^k. Advantage of such an architecture is that it supports implementation of so many logic functions, however the disadvantage is unusually large number of memory cells required to implement such a logic block in case number of inputs is large.

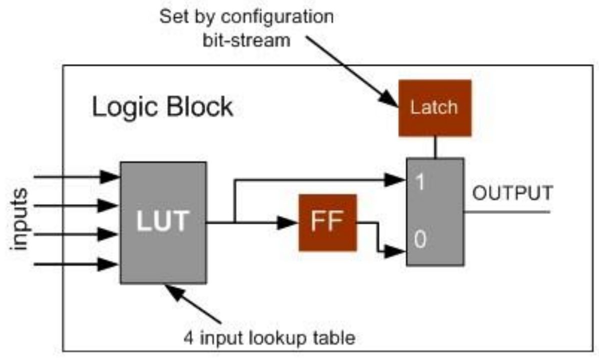


Figure 22 A 4-input LUT based implementation of logic block

LUT based design provides for better logic block utilization. A k-input LUT based logic block can be implemented in number of different ways with trade off between performance and logic density. An n-LUT can be shown as a direct implementation of a function truth-table. Each of the latch hold's the value of the function corresponding to one input combination. For Example: 2-LUT can be used to implement 16 types of functions like AND , OR, A+not B .... Etc.

5.3.1.1.1 Interconnects

A wire segment can be described as two end points of an interconnect with no programmable switch between them. A sequence of one or more wire segments in an FPGA can be termed as a track. Typically an FPGA has logic blocks, interconnects and switch blocks (Input/Output blocks). Switch blocks lie in the periphery of logic blocks and interconnect. Wire segments are connected to logic blocks through switch blocks. Depending on the required design, one logic block is connected to another and so on.

5.3.1.1.2 FPGA Design Flow

In this part of report we are going to have a short intro on FPGA design flow. A simplified version of design flow is given in the flowing diagram.

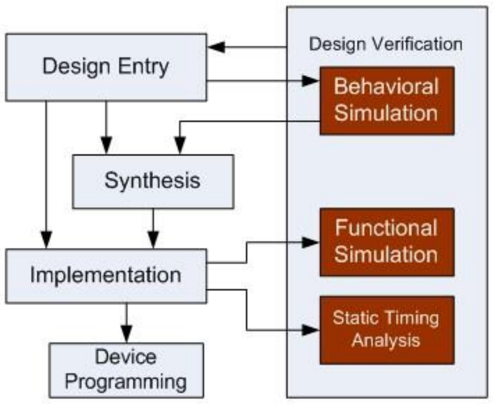


Figure 23 FPGA Design Flow

5.3.1.2 Design Entry

There are different techniques for design entry. Schematic based, Hardware Description Language and combination of both etc.. Selection of a method depends on the design and designer. If the designer wants to deal more with Hardware, then Schematic entry is the better choice. When the design is complex or the designer thinks the design in an algorithmic way then HDL is the better choice. Language based entry is faster but lag in performance and density.

HDLs represent a level of abstraction that can isolate the designers from the details of the hardware implementation. Schematic based entry gives designers much more visibility into the hardware. It is the better choice for those who are hardware oriented. Another method but rarely used is state-machines. It is the better choice for the designers who think the design as a series of states. But the tools for state machine entry are limited. In this documentation we are going to deal with the HDL based design entry.

5.3.1.3 Synthesis

The process which translates VHDL or Verilog code into a device netlist formate. i.e a complete circuit with logical elements(gates, flip flops, etc…) for the design. If the design contains more than one sub designs, ex. to implement a processor, we need a CPU as one design element and RAM as another and so on, then the synthesis process generates netlist for each design element Synthesis process will check code syntax and analyze the hierarchy of the design which ensures that the design is optimized for the design architecture, the designer has selected. The resulting netlist(s) is saved to an NGC(Native Generic Circuit) file (for Xilinx® Synthesis Technology (XST)).

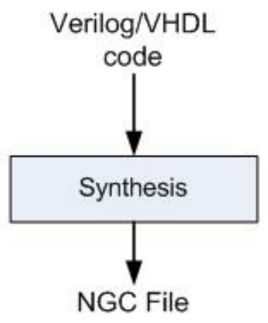


Figure 24 Synthesis Process

5.3.1.4 Implementation

This process consist's a sequence of three steps

- Translate
- Map
- Place and Route

5.3.1.4.1 Translate:

Process combines all the input netlists and constraints to a logic design file. This information is saved as a NGD (Native Generic Database) file. This can be done using NGD Build program. Here, defining constraints is nothing but, assigning the ports in the design to the physical elements (ex. pins, switches, buttons etc) of the targeted device and specifying time requirements of the design. This information is stored in a file named UCF (User Constraints File). Tools used to create or modify the UCF are PACE, Constraint Editor etc.

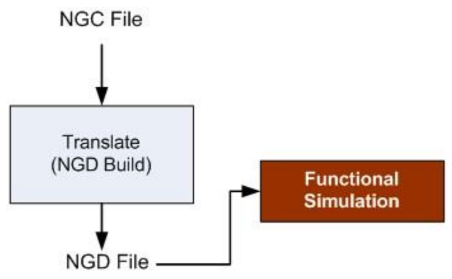


Figure 25 FPGA Translate

5.3.1.4.2 Map

Process divides the whole circuit with logical elements into sub blocks such that they can be fit into the FPGA logic blocks. That means map process fits the logic defined by the NGD file into the targeted FPGA elements (Combinational Logic Blocks (CLB), Input Output Blocks (IOB)) and generates an NCD (Native Circuit Description) file which physically represents the design mapped to the components of FPGA. MAP program is used for this purpose.

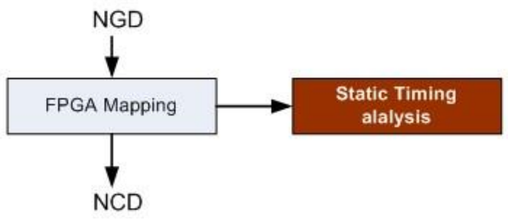


Figure 26 FPGA map

5.3.1.4.3 Place and Route:

PAR program is used for this process. The place and route process places the sub blocks from the map process into logic blocks according to the constraints and connects the logic blocks. Ex. if a sub block is placed in a logic block which is very near to IO pin, then it may save the time but it may effect some other constraint. So trade off between all the constraints is taken account by the place and route process.

The PAR tool takes the mapped NCD file as input and produces a completely routed NCD file as output. Output NCD file consists the routing information.

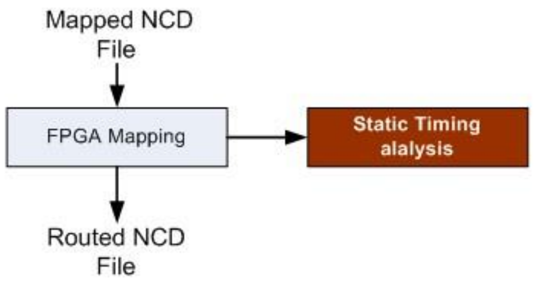


Figure 27 FPGA Place and route

5.3.1.5 Device Programming:

Now the design must be loaded on the FPGA. But the design must be converted to a format so that the FPGA can accept it. BITGEN program deals with the conversion. The routed NCD file is then given to the BITGEN program to generate a bit stream (a .BIT file) which can be used to configure the target FPGA device. This can be don e using a cable. Selection of cable depends on the design.

5.4. Synthesis Result

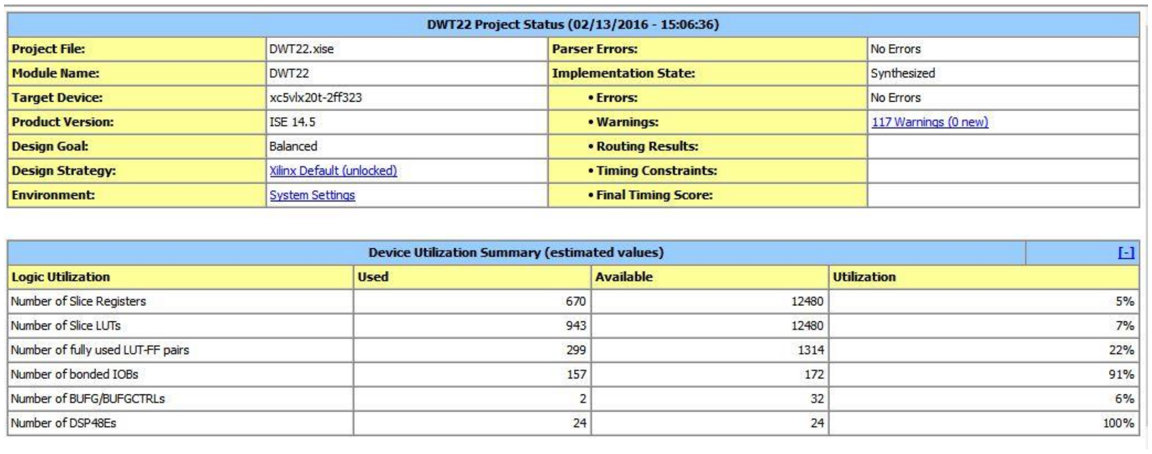
The developed DWT, is simulated and verified their functionality. Once the functional verification is done, the RTL model is taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level netlist mapped to a specific technology library.

The design of DWT is synthesized and its results were analyzed as follows.

5.4.1.1.1 DWT Synthesis Result

This device utilization includes the following.

- Logic Utilization
- Logic Distribution
- Total Gate count for the Design

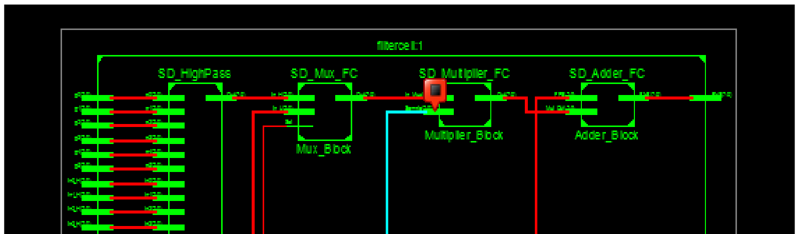


DWT22 Project Status (02/13/2016 - 15:06:36)			
Project File:	DWT22.xise	**Parser Errors:**	No Errors
Module Name:	DWT22	**Implementation State:**	Synthesized
Target Device:	xc5vlx20t-2ff323	• **Errors:**	No Errors
Product Version:	ISE 14.5	• **Warnings:**	117 Warnings (0 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	670	12480	5%
Number of Slice LUTs	943	12480	7%
Number of fully used LUT-FF pairs	299	1314	22%
Number of bonded IOBs	157	172	91%
Number of BUFG/BUFGCTRLs	2	32	6%
Number of DSP48Es	24	24	100%

Figure 28 Synthesis Report

5.5. RTL Schematic



\

Figure 29 Filter Cell Architecture

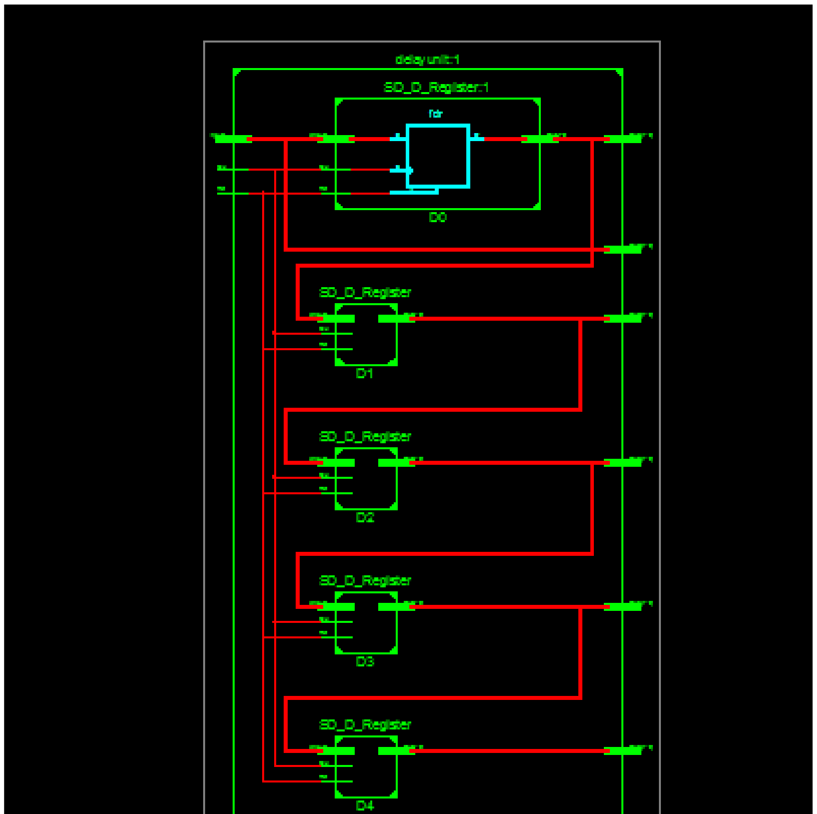


Figure 30 Register Bank Unit.

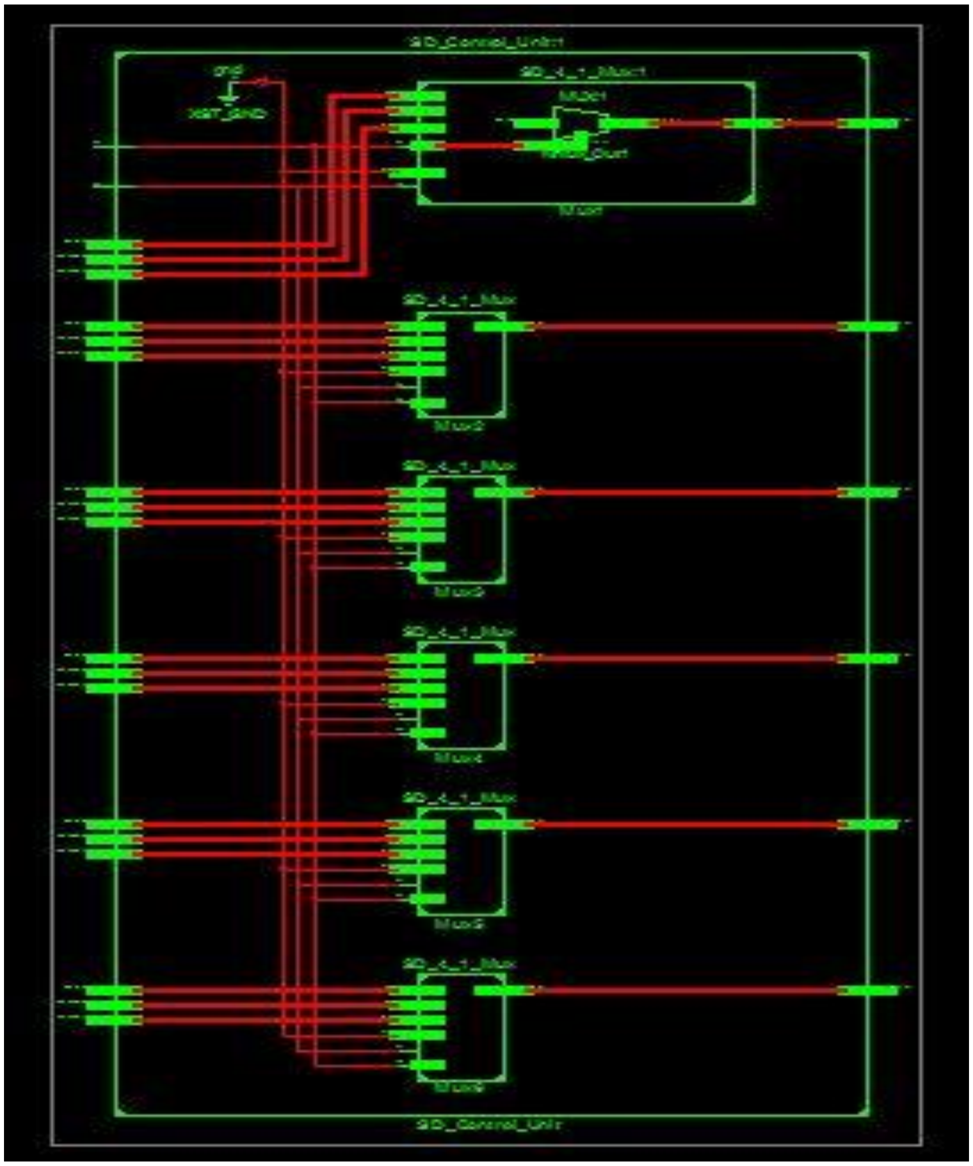


Figure 31 Control Unit Schematic

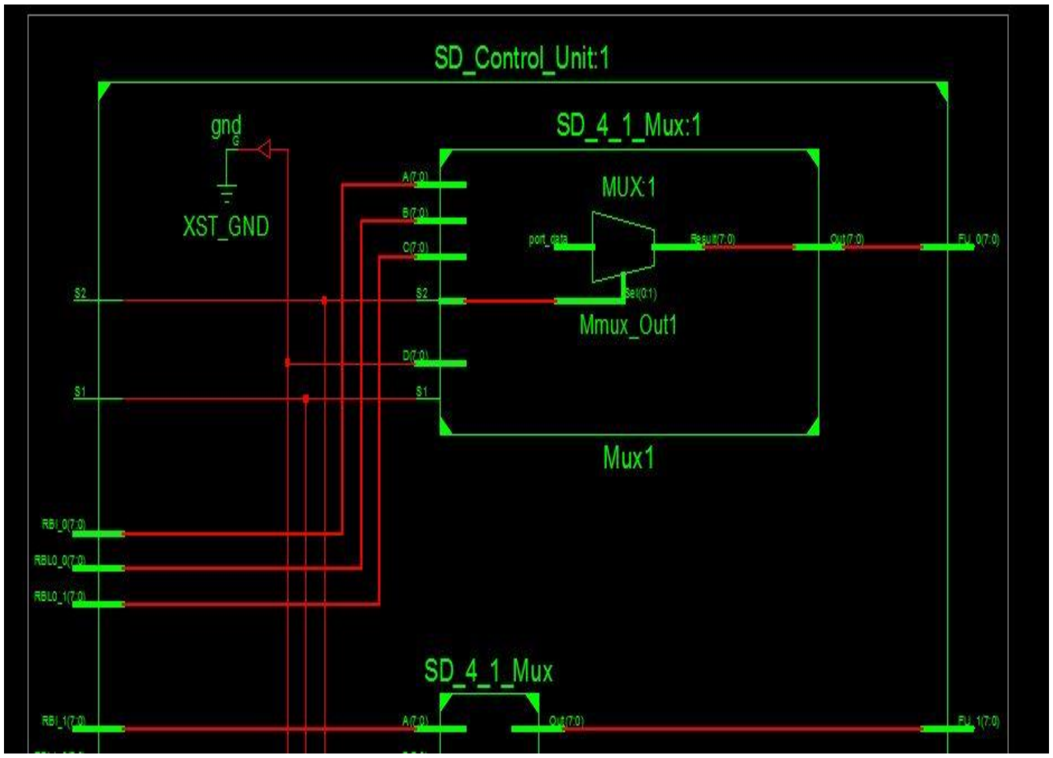


Figure 32 Control Unit One block

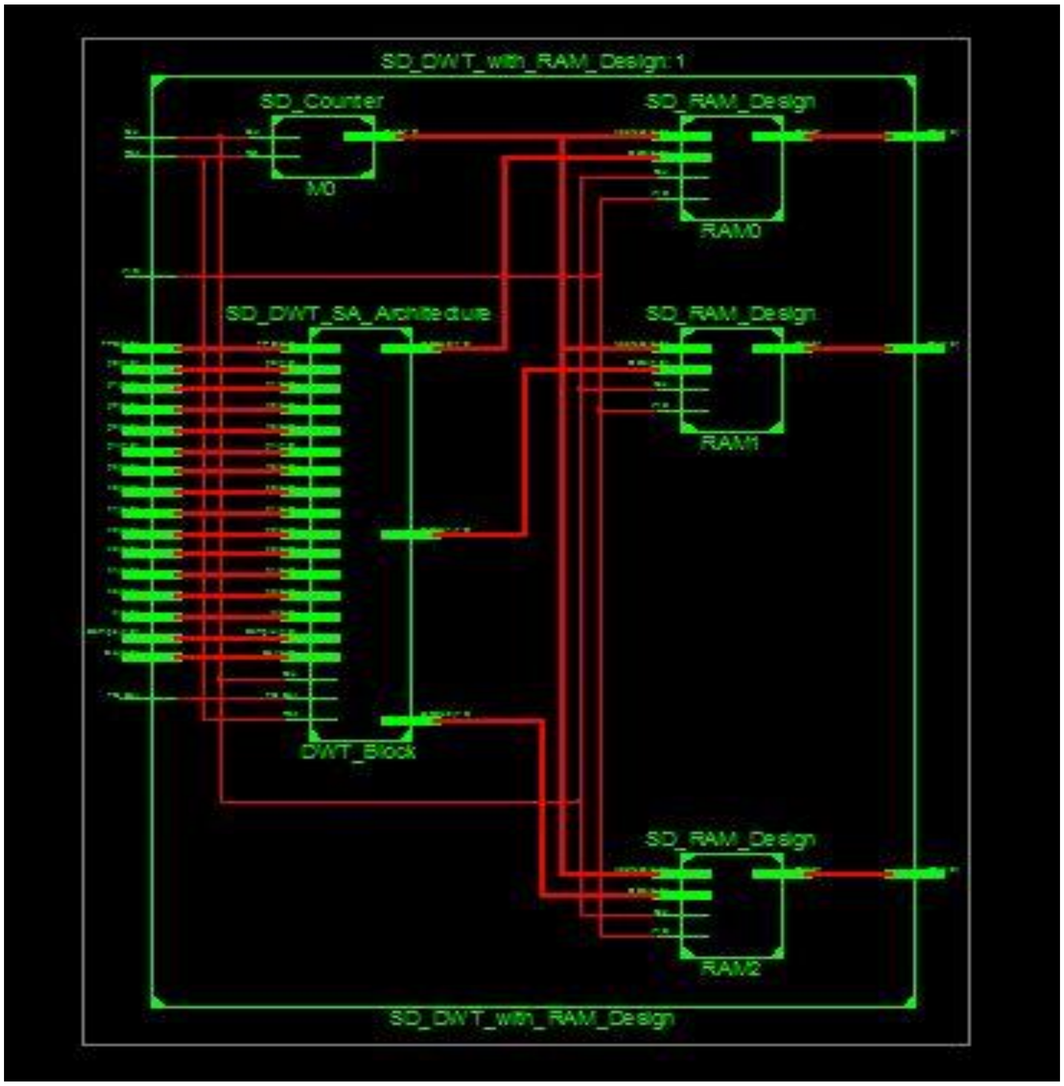


Figure 33 SA-DWT with RAM memory

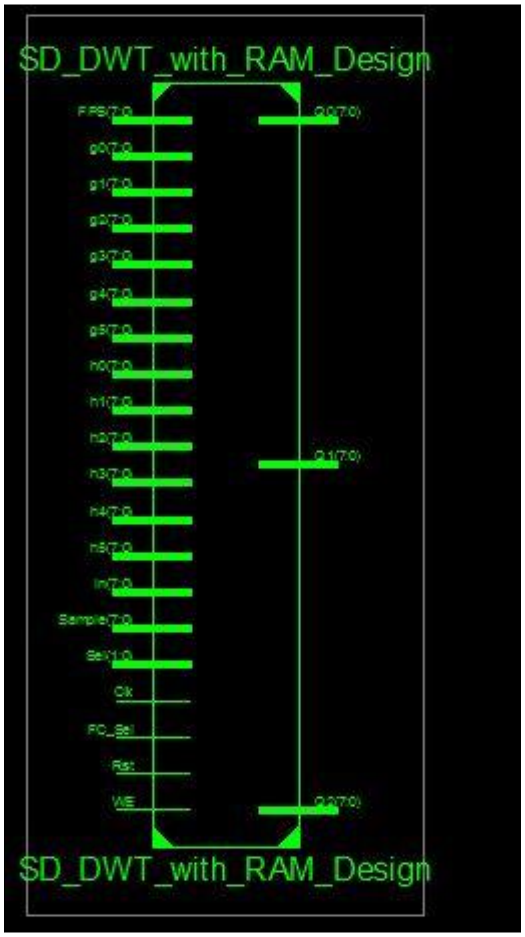


Figure 34 Top level View of SA-DWT

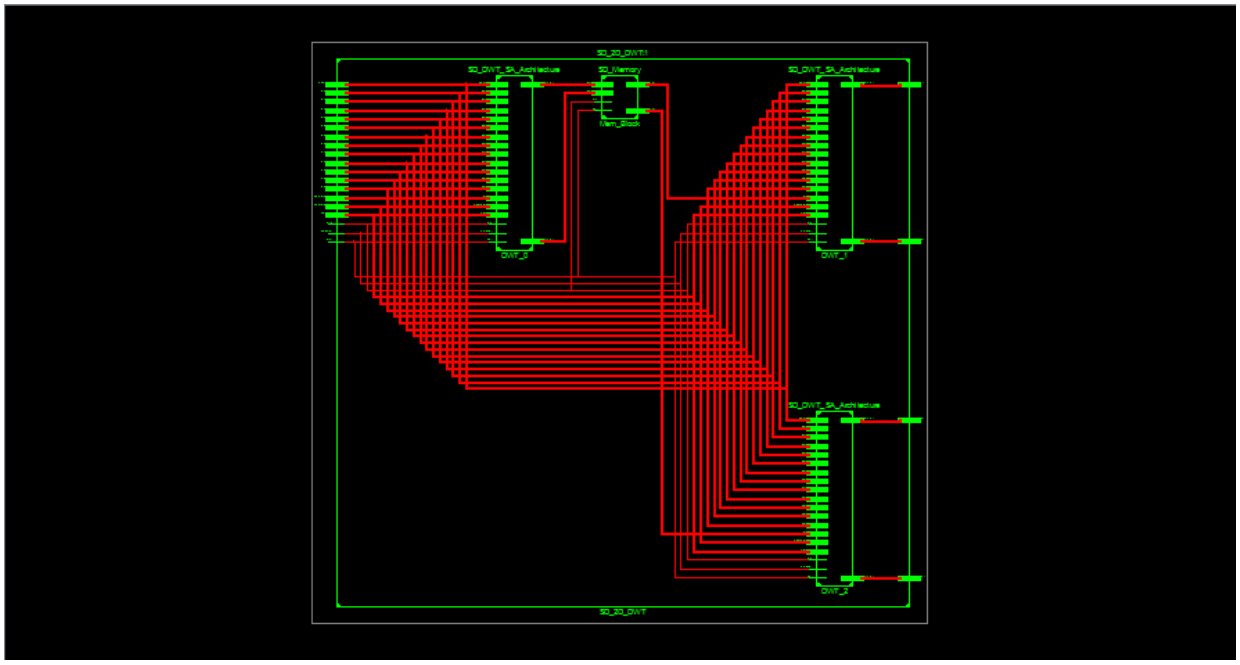


Figure 35 2D SA-DWT

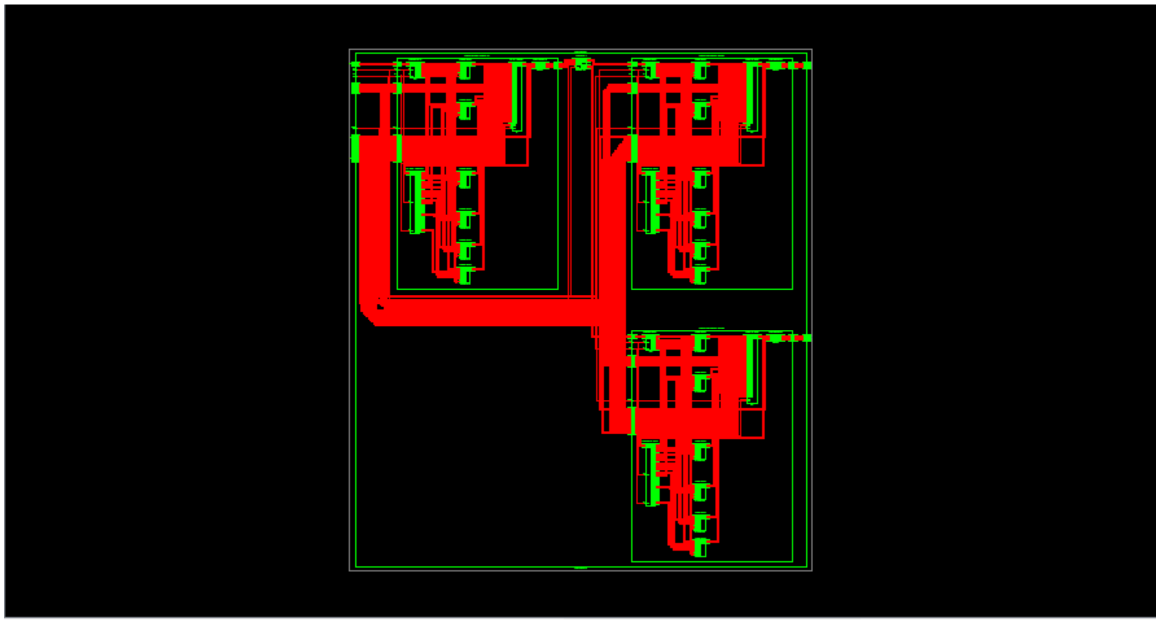


Figure 36 SA-DWT RTL Schematic(1)

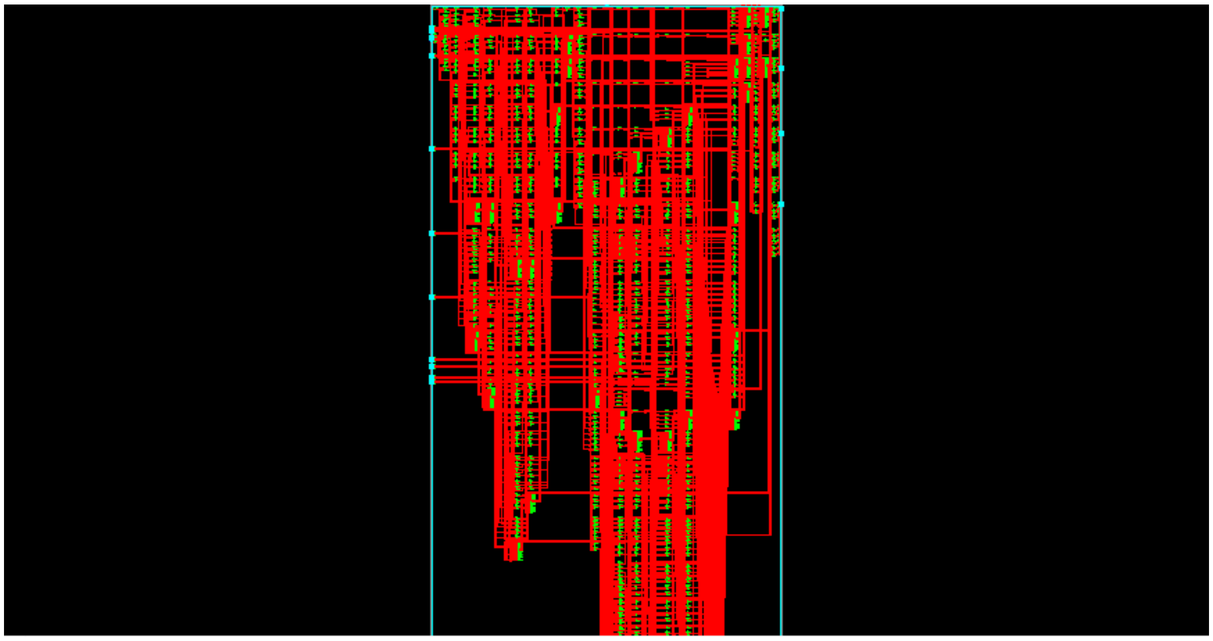
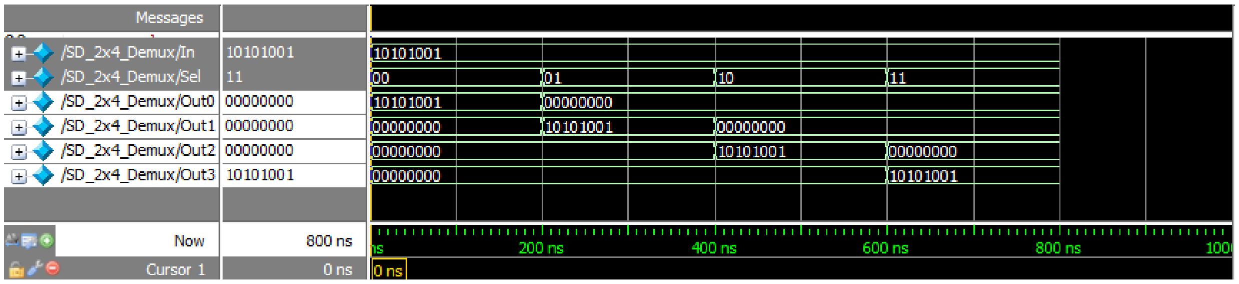
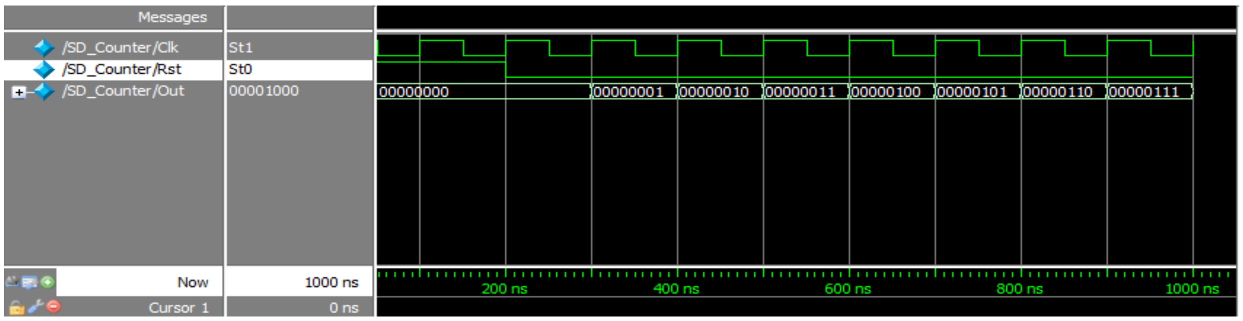


Figure 37 RTL Schematic



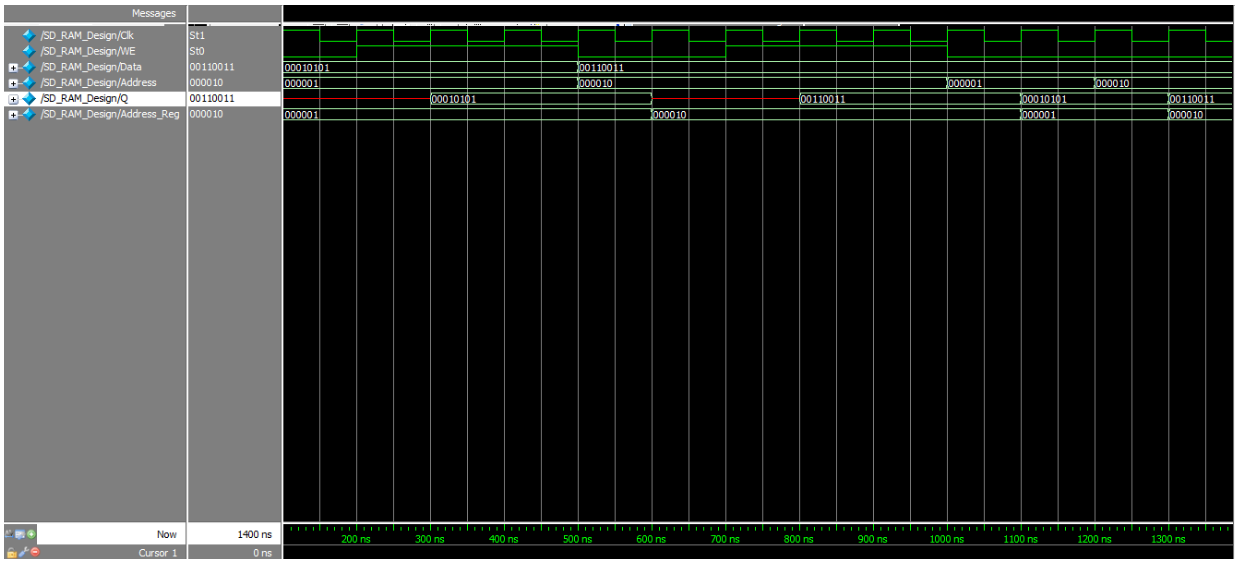
```
Messages
/SD_2x4_Demux/In    10101001   10101001
/SD_2x4_Demux/Sel   11         00 | 01 | 10 | 11
/SD_2x4_Demux/Out0  00000000   10101001 | 00000000
/SD_2x4_Demux/Out1  00000000   00000000 | 10101001 | 00000000
/SD_2x4_Demux/Out2  00000000   00000000 | 10101001 | 00000000
/SD_2x4_Demux/Out3  10101001   00000000 | 10101001
Now       800 ns    ns  200 ns  400 ns  600 ns  800 ns  100
Cursor 1  0 ns      0 ns
```

Figure 38 2X4 Demux(level 1, level 2, level 3 Results)



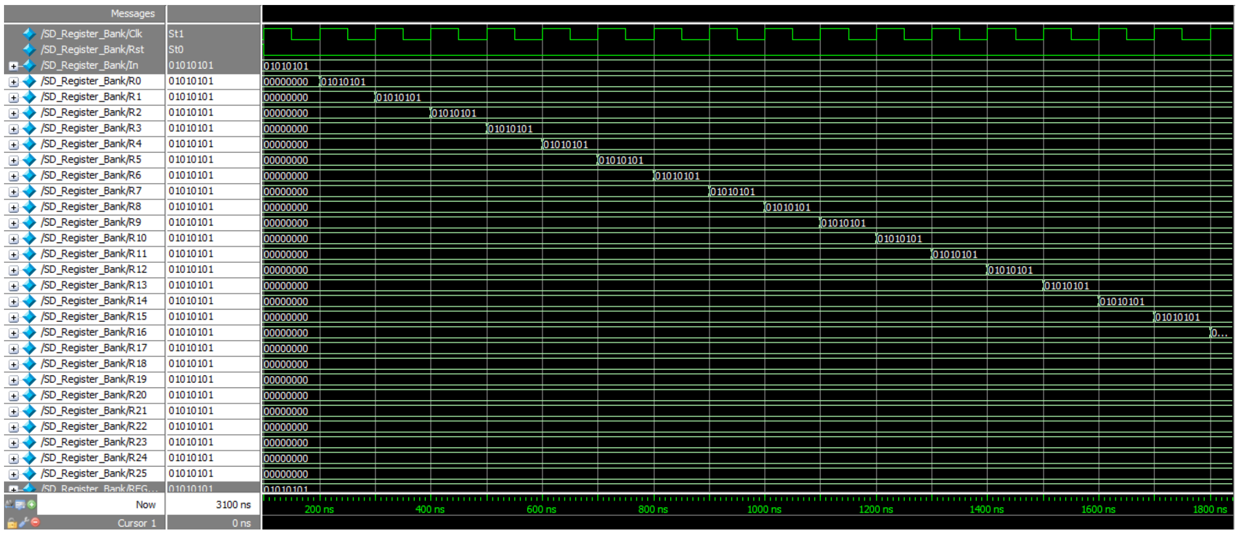
```
Messages
/SD_Counter/Clk   St1
/SD_Counter/Rst   St0
/SD_Counter/Out   00001000   00000000 | 00000001 | 00000010 | 00000011 | 00000100 | 00000101 | 00000110 | 00000111
Now       1000 ns   200 ns  400 ns  600 ns  800 ns  1000 ns
Cursor 1  0 ns
```

Figure 39 Counter Simulation Result



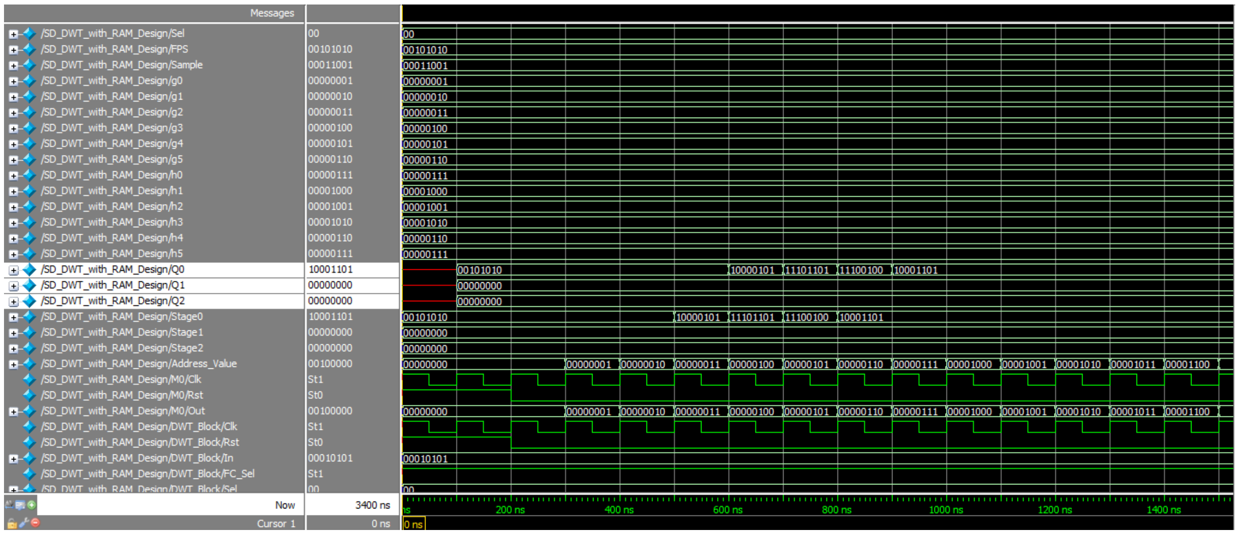
```
Messages
/SD_RAM_Design/Clk           St1
/SD_RAM_Design/WE            St0
/SD_RAM_Design/Data          00110011
/SD_RAM_Design/Address       000010
/SD_RAM_Design/Q             00110011
/SD_RAM_Design/Address_Reg   000010
Now                          1400 ns
Cursor 1                     0 ns
```

Figure 40 Memory Block Simulation Result



```
Messages
/SD_Register_Bank/Clk     St1
/SD_Register_Bank/Rst     St0
/SD_Register_Bank/In      01010101
/SD_Register_Bank/R0      01010101
/SD_Register_Bank/R1      01010101
/SD_Register_Bank/R2      01010101
/SD_Register_Bank/R3      01010101
/SD_Register_Bank/R4      01010101
/SD_Register_Bank/R5      01010101
/SD_Register_Bank/R6      01010101
/SD_Register_Bank/R7      01010101
/SD_Register_Bank/R8      01010101
/SD_Register_Bank/R9      01010101
/SD_Register_Bank/R10     01010101
/SD_Register_Bank/R11     01010101
/SD_Register_Bank/R12     01010101
/SD_Register_Bank/R13     01010101
/SD_Register_Bank/R14     01010101
/SD_Register_Bank/R15     01010101
/SD_Register_Bank/R16     01010101
/SD_Register_Bank/R17     01010101
/SD_Register_Bank/R18     01010101
/SD_Register_Bank/R19     01010101
/SD_Register_Bank/R20     01010101
/SD_Register_Bank/R21     01010101
/SD_Register_Bank/R22     01010101
/SD_Register_Bank/R23     01010101
/SD_Register_Bank/R24     01010101
/SD_Register_Bank/R25     01010101
/SD_Register_Bank/REG...  01010101
Now                       3100 ns
Cursor 1                  0 ns
```

Figure 41Register Bank Block Simulation Result



```
Messages
/SD_DWT_with_RAM_Design/Sel                  00         00
/SD_DWT_with_RAM_Design/FPS                  00101010   00101010
/SD_DWT_with_RAM_Design/Sample               00011001   00011001
/SD_DWT_with_RAM_Design/g0                   00000001   00000001
/SD_DWT_with_RAM_Design/g1                   00000010   00000010
/SD_DWT_with_RAM_Design/g2                   00000011   00000011
/SD_DWT_with_RAM_Design/g3                   00000100   00000100
/SD_DWT_with_RAM_Design/g4                   00000101   00000101
/SD_DWT_with_RAM_Design/g5                   00000110   00000110
/SD_DWT_with_RAM_Design/h0                   00000111   00000111
/SD_DWT_with_RAM_Design/h1                   00001000   00001000
/SD_DWT_with_RAM_Design/h2                   00001001   00001001
/SD_DWT_with_RAM_Design/h3                   00001010   00001010
/SD_DWT_with_RAM_Design/h4                   00000110   00000110
/SD_DWT_with_RAM_Design/h5                   00000111   00000111
/SD_DWT_with_RAM_Design/Q0                   10001101   00101010 10000101 11101101 11100100 10001101
/SD_DWT_with_RAM_Design/Q1                   00000000   00000000
/SD_DWT_with_RAM_Design/Q2                   00000000   00000000
/SD_DWT_with_RAM_Design/Stage0               10001101   00101010 10000101 11101101 11100100 10001101
/SD_DWT_with_RAM_Design/Stage1               00000000   00000000
/SD_DWT_with_RAM_Design/Stage2               00000000   00000000
/SD_DWT_with_RAM_Design/Address_Value        00100000   00000000 00000001 00000010 00000011 00000100 00000101 00000110 00000111 00001000 00001001 00001010 00001011 00001100
/SD_DWT_with_RAM_Design/M0/Clk               St1
/SD_DWT_with_RAM_Design/M0/Rst               St0
/SD_DWT_with_RAM_Design/M0/Out               00100000   00000000 00000001 00000010 00000011 00000100 00000101 00000110 00000111 00001000 00001001 00001010 00001011 00001100
/SD_DWT_with_RAM_Design/DWT_Block/Clk        St1
/SD_DWT_with_RAM_Design/DWT_Block/Rst        St0
/SD_DWT_with_RAM_Design/DWT_Block/In         00010101   00010101
/SD_DWT_with_RAM_Design/DWT_Block/FC_Sel     St1
/SD_DWT_with_RAM_Design/DWT_Block/Sel        00         00
                                    Now      3400 ns    ns  200 ns  400 ns  600 ns  800 ns  1000 ns  1200 ns  1400 ns
                                    Cursor 1    0 ns    0 ns
```

Figure 42 2D SA-DWT Simulation Result

CHAPTER.7

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

Basically the medical images need more accuracy without loosing of information. The Discrete Wavelet Transform (DWT) was based on time-scale representation, which provides efficient multi-resolution. filter give lossless mode of information. A more efficient approach to lossless whose coefficients are exactly represented by finite precision numbers allows for truly lossless encoding.

This work ensures that the image pixel values given to the DWT process which gives the high pass and low pass coefficients of the input image. The simulation results of DWT were verified with the appropriate test cases. Once the functional verification is done, discrete wavelet transform is synthesized by using Xilinx tool. Hence it has been analyzed that the discrete wavelet transform (DWT) operates at a maximum clock frequency of 99.197 MHz respectively.

6.2. Future Scope

- This work can be extended in order to increase the accuracy by increasing the level of transformations.
- This can be used as a part of the block in the full fledged application, i.e., by using these DWT, the applications can be developed such as compression, watermarking, etc.

CHAPTER.8

7. REFERENCES

[1] Chih-Hsien Hsia, Jen-Shiun Chiang, "Memory-Efficient Hardware Architecture of 2-D Dual-Mode Lifting-Based Discrete Wavelet Transform" IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 4,pp 671-683, 2013.

[2] Yusong Hu and Ching Chuen Jong "A Memory-Efficient Scalable Architecture for Lifting-Based Discrete Wavelet Transform" IEEE Transactions on Circuits and Systems II, Exp Briefs, vol. 60, no. 8,pp 4975-4987, 2013

[3] Basant Kumar Mohanty and P.K.Meher "Memory-Efficient High-Speed Convolutionbased Generic Structure for Multilevel 2-D DWT" IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp 353-363,2013.

[4] Chao cheng and K K Parhi "High Speed VLSI Implementation OF 2-D DWT" IEEE Transactions on Signal Processing, vol. 56,no.1,pp 385-391, 2008.

[5] Qing sin and Jiang Jiang "A Reconfigurable Architecture for 1-D and 2-D Discrete Wavelet Transform" IEEE Transactions on Field Programmable Custom Computing Machines,vol.3,pp 81-84,2013

[6] Po-Cheng Wu and Liang-Gee Chen "An Efficient Architecture for Two-Dimensional Discrete Wavelet Transform" IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, no. 4,pp 536-544, 2001.

[7] Xin Tian, Lin Wu, Yi-Hua Tan, and Jin-Wen Tian "Efficient Multi-Input/MultiOutput VLSI Architecture for Two-Dimensional Lifting-Based Discrete Wavelet Transform" IEEE Transactions on Computers, vol. 60, no. 8,pp 1207-1208,2011

[8] Yeong-Kang Lai, Member ,Lien-Fei Chen, and Yui-Chih Shih " A High-Performance and Memory-Efficient VLSI Architecture with Parallel Scanning Method for 2-D Lifting-Based Discrete Wavelet Transform" IEEE Transactions on Consumer Electronics, vol. 55, no. 2,pp 400-407, 2009

[9] Chao-Tsung Huang, Po-Chih Tseng, and Liang-Gee Chen "Flipping Structure: An Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Transform" IEEE Transactions on Signal Processing, vol. 52, no. 4,pp 1080-1089, 2004

[10] Francescomaria Marino, David Guevorkian, and Jaakko T. Astola "Highly Efficient High-Speed/Low-Power Architectures for the 1-D Discrete Wavelet Transform" IEEE Transactions on Circuits and Systems-II Analog and Digital Signal Processing, vol. 47, no. 12,pp 1492-1502,2000

[11] http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html ; the Wavelet Tutorial by Robi Polikar.