Scholars Journal of Engineering and Technology

OPEN ACCESS

Abbreviated Key Title: Sch J Eng Tech
ISSN 2347-9523 (Print) | ISSN 2321-435X (Online)
Journal homepage: https://saspublishers.com

FPGA Implementation of Efficient CDF-9/7 Discrete Wavelet Transform

Roopa K. C[1*], Chetan S[2]

[1]M. Tech Student, Department of Electronics and Communication, Dr. Ambedkar Institute of Technology, Bengaluru-560056, Karnataka, India
[2]Assistant Professor Department of Electronics and Communication, Dr. Ambedkar Institute of Technology, Bengaluru-560056, Karnataka, India

DOI: 10.36347/sjet.2023.v11i10.001 | **Received:** 22.08.2023 | **Accepted:** 30.09.2023 | **Published:** 03.10.2023

***Corresponding author:** Roopa K. C
M. Tech Student, Department of Electronics and Communication, Dr. Ambedkar Institute of Technology, Bengaluru-560056, Karnataka, India

Abstract | **Original Research Article**

Architecture CDF-5/3 DWT is utilized to derive CDF-9/7 DWT. Because of the prevalence of fractional parts in the generic CDF 9/7 architecture and the occurrence of truncation errors, the design is difficult to execute. Since the CDF-5/3 DWT filter allows for the employment of merely a shifter, attempted to design the CDF-9/7 DWT filter in this manner so that the truncation error is as little as possible, if not eliminated.

Keywords: CDF-5/3 DWT, CDF 9/7, Truncation error, Employment of merely

1. INTRODUCTION

Almost every modern technology relies on methods of digital signal processing. Any application using digital signal processing relies on accurate analysis of the input signal's stated properties. Once natural signals are sampled and held in digital form, they exist only in the temporal domain. However, several computational challenges make it difficult to extract most of the properties from time domain data. Domain approaches such as frequency domain, z-domain, and s-domain have been developed to address this issue. Frequency domain processing makes detection and processing simpler since it uses most of the available processing domains. Because of this, it is the most often used DSP application domain.

Several methods exist for transitioning from the time domain to the frequency domain. DFT, DWT, DCT, etc., are some of the most commonly used methods. The Discrete Wavelet Transform (DWT) is a famous domain transform method with many benefits over competitors.

The primary drawbacks of DWT are the need for a sizable multiplier and a massive quantity of storage to keep track of the values for each twiddle factor. The multiplier and memory components scale with the DWT's number of points. This meant the building needed plenty of room yet could run at a snail's pace. The discrete wavelet transform, a novel method developed to solve these issues, is also utilized for picture compression and feature extraction.

2. Needed computer systems and applications

Hardware and software are chosen for the project. Spartan 6 FPGA (ATLYS) hardware from Xilinx. It is common practice to use Xilinx ISE 14.5 and the MATLAB system generator to write HDL and VHDL code, respectively.

2.1. Xilinx Spartan-6 FPGA

The Xilinx Spartan-6 FPGA Project Board is a digital system development board with a Spartan-6 FPGA, 4MB of external non-volatile memory, and sufficient I/O devices and external connectors to interface a range of digital applications.

Due to their ability to begin system software development concurrently with hardware, enable system performance simulations at a very early stage of the development, and permit various system trials and design iterations before finalizing the system architecture, FPGAs have a remarkable role in embedded system development.

2.2. Xilinx ISE 14.5

A software program used for circuit design and synthesis is called Xilinx ISE. Following the simulation of the RTL code, RTL diagrams are produced. The synthesis, design, and timing analysis of circuits are some of the key functions of Xilinx ISE. Spartan 6 is supported by the ISE design suite. Both Linux and Windows 10 support the ISE design suite. It is used in

this project to write the code for a highly detailed language.

2.3. System Generation using MATLAB

MATLAB, short for "Mathematical Arrays and Matrix Laboratory," is a suite of programming languages and a desktop environment for doing iterative design and analysis. The MATLAB add-ons have been meticulously crafted, thoroughly examined, and meticulously documented. As will be shown, toolkits are essential for the implementation of the proposed specific solutions. MATLAB's features include ICC color management, multidimensional image processing, the presentation of pictures and videos through the sequential application of DCT and FFT transforms, and the processing and display of images of larger sizes to predict histogram and entropy values. MATLAB-created software is mostly utilized for testing purposes.

2.4. Architectures

Most of the new FPGA architectures consist of configurable logic blocks (CLBs)/logic array blocks (LABs), I/O pads and configurable routing logics. The I/O pads are mainly used to connect input/output to/form the FPGA chip. The I/O pads consist of some buffer cells for good voltage level and minimize fan-in/fan-out problem.

The programmable routing logic is one of the most complicated designs in whole FPGA. The interconnect lines inside the FPGA is made in such way that any line can works as both input/output/in out line. To make that different techniques are used. One such technique is shown in Fig 1.

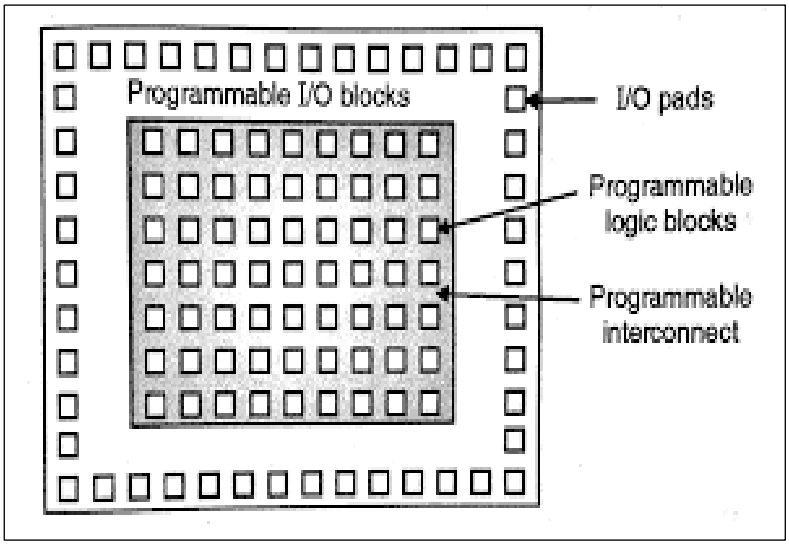

Fig 1: (a) Basic FPGA block

Comprehensive FPGA design is presented.

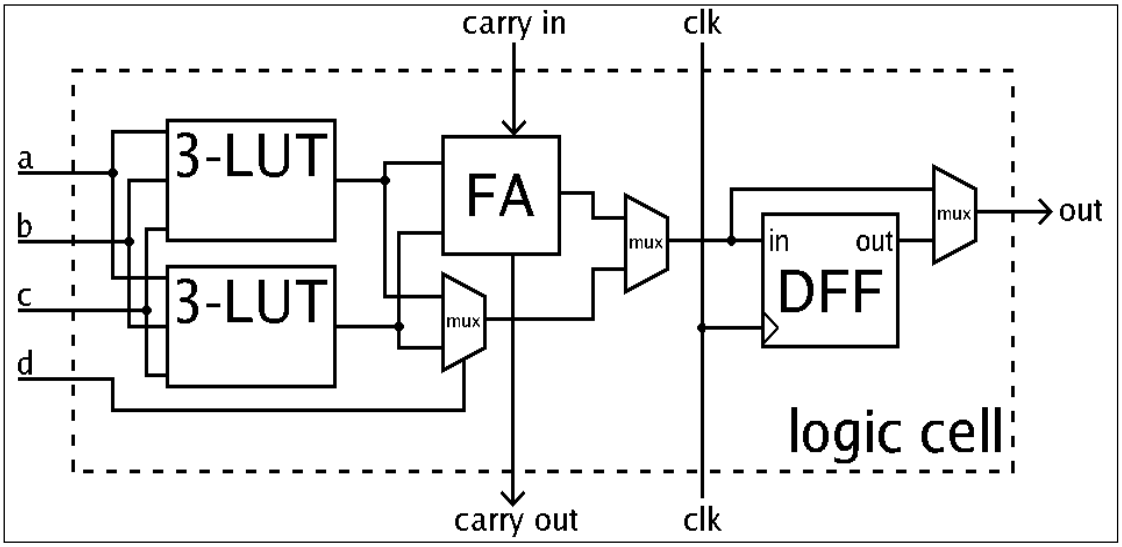

Fig 1: (b) FPGA's Primitive Structure

2.5 Design and Implementation

The flowchart illustrates the process by which an FPGA is developed and put into operation. Fig 2 *FPGA* flow Chart.

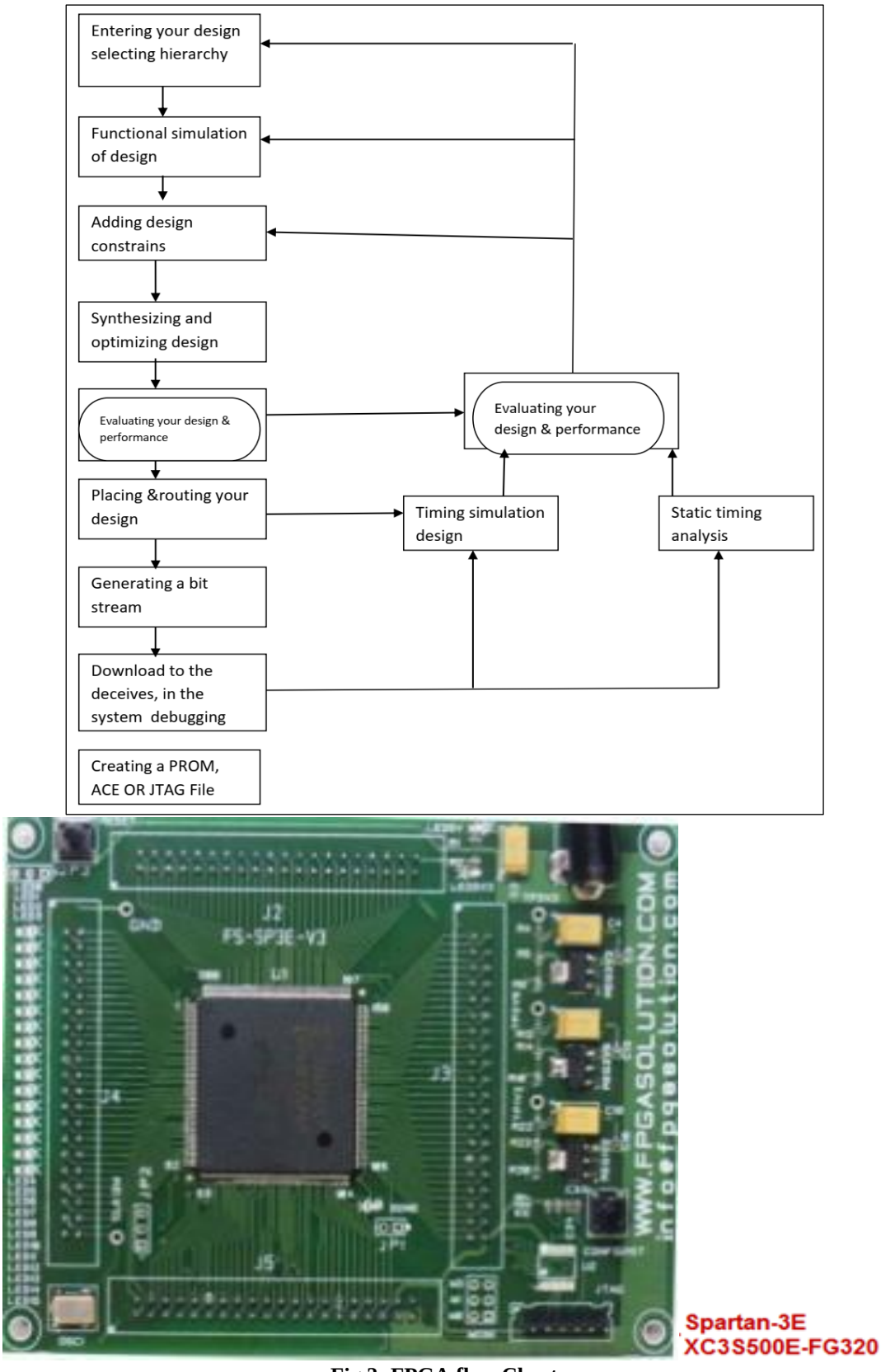

Fig 2: FPGA flow Chart

2.6 Basic Structure of XILNX FPGA

XILINX FPGA's fundamental internal architecture is seen in Fig.3 below. The logic blocks, I/O ports, and the interface of this FPGA are all configurable. The programmable is created by putting the whole setup into RAM.

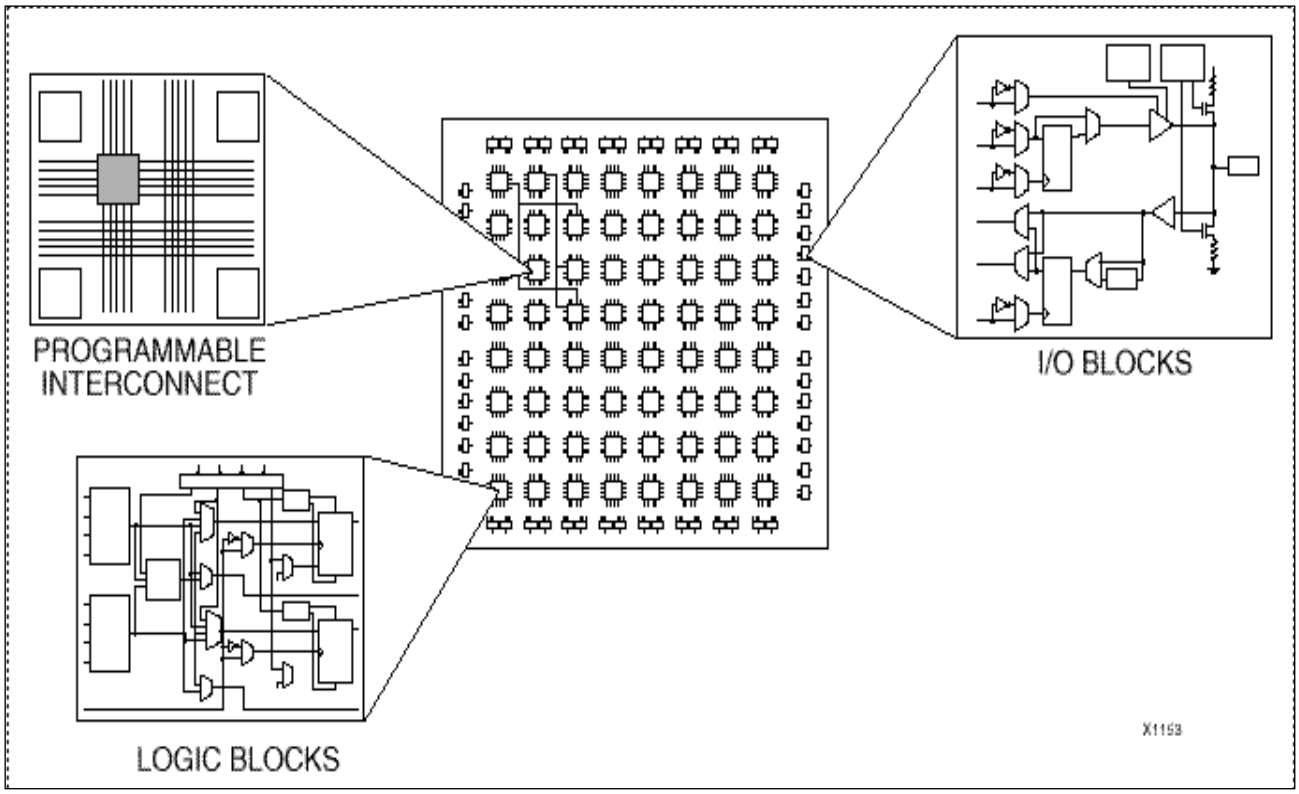

Fig 3: FPGA Fundamentals Using XILINX

2.7 Xilinx ATLYS FPGA

ATLYS FPGA board is a Xilinx Spartan-6 FPGA (xc6slx45-2csg324). Fig 4 is a schematic representation of the functionality of this board. This is the bare minimum needed for any DSP-based program. This board is ideal for DSP implementations since it has all the necessary internal peripherals.

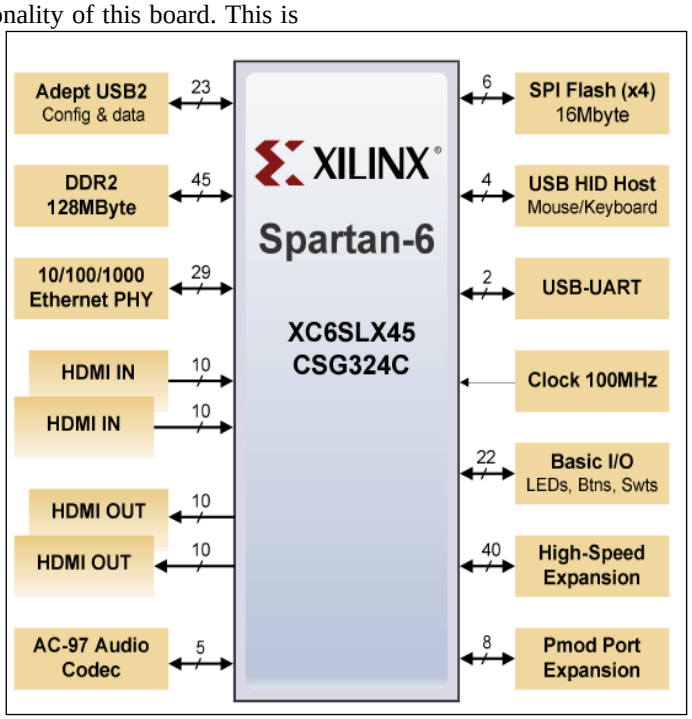

Fig 4: ATLYS FPGA Functional Block Diagram

2.8 Generator System Fundamentals

Xilinx's system generator is a tool that lets you use the Simulink IDE for FPGA layout. In the Simulink modeling system, designs are stored as blocks. Human interaction is required at any stage of the FPGA implementation process. Registers, multipliers, and adders are just a few of the more than eighty DSP building blocks provided by Xilinx in their DSP block set for Simulink. It also provides:

- A unified platform for FPGA design.
- Facilitating the integration of RTL, Simulink.
- The associated simulation and application environments.

The system generator maintains a black box that allows RTL to be imported into Simulink and co-simulated using modelsim, Xilinx, or ISE. Fig 5: shows the flow System generator flow design.

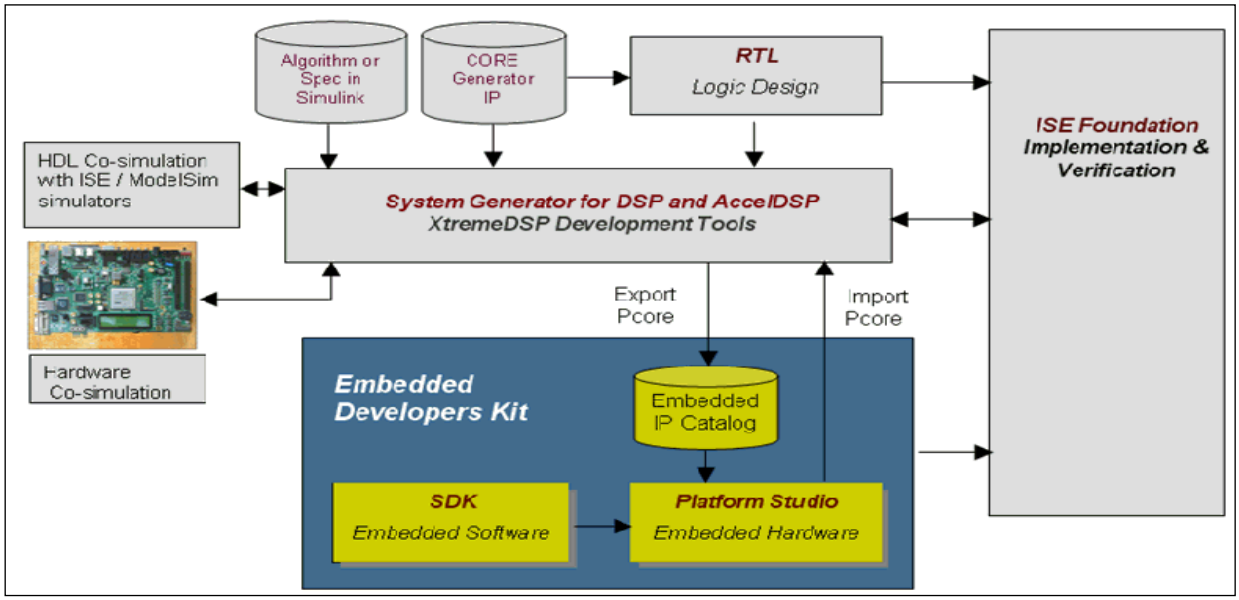

Fig 5: System generator flow design

3. METHODOLOGY

3.1 1D-DWT

At the next-to-lowest resolution level, the 1D-DWT iteratively decomposes the input signal So(n) in approximation and detail. The approximate signal at level n is denoted by Si (n), whereas the exact signal at level n is denoted by Wi (n). The approximate and fine-tune the signal at the 'i+1' level using these values.

$$si + 1(n) = \Sigma(k = 0)^{(L-1)g(k)si}(2n - k)$$
$$wi + 1(n) = \Sigma(k = 0)^{(L-1)h(k)si}(2n - k)$$

The low pass and high pass filter coefficients, g(K) and h(K), and the filter length, L, are defined as follows. Fig 6 below depicts the three-level 1D DWT.

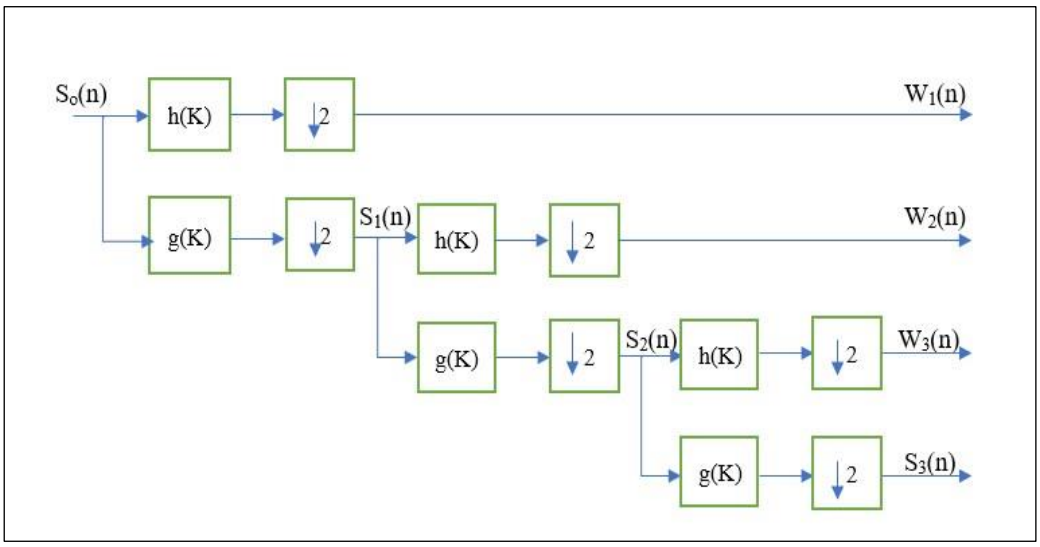

Fig 6: Three 1D-DWT Levels

3.2. 2D-DWT

In many fields, including image compression, computer vision, and multi-resolution analysis, use the 2D-DWT. Separately and non-separable 2D filters are used here. Separatability is defined as the ability to write a 2D filter f(n1, n2) as f(n1, n2) = f1(n1).Whereas f1(n1) and f2(n2) are 1D filters, f2(n2) is a 2D filter. Separate 2D discrete wavelet transforms may break down an approximation picture Si(n1, n2) into the original image and three details.

$$Si+1(n1, n2) = \sum\_ (K1=0) \wedge (L-1) \sum\_ (K2=0) \wedge (L-1)[g(K1)]\ g(K2)\ Si(2n1 – K1, 2n2 – K2)$$

$$W1i+1\ (n1, n2) = \sum\_(K1=0)\wedge(L-1) \sum\_(K2=0) \wedge (L-1)\ [g(K1)]\ h(K2)\ Si(2n1 – K1, 2n2 – K2)$$

1D wavelet filters G(Z) and H(Z) are available. Si+1(n1, n2), with its lesser resolution, is close to Si (n1, n2). This estimate is derived from Si (n1, n2) by low pass filtering and decimation by '2' along the row and column. The one level 2D DWT is shown in the Fig 7 below.

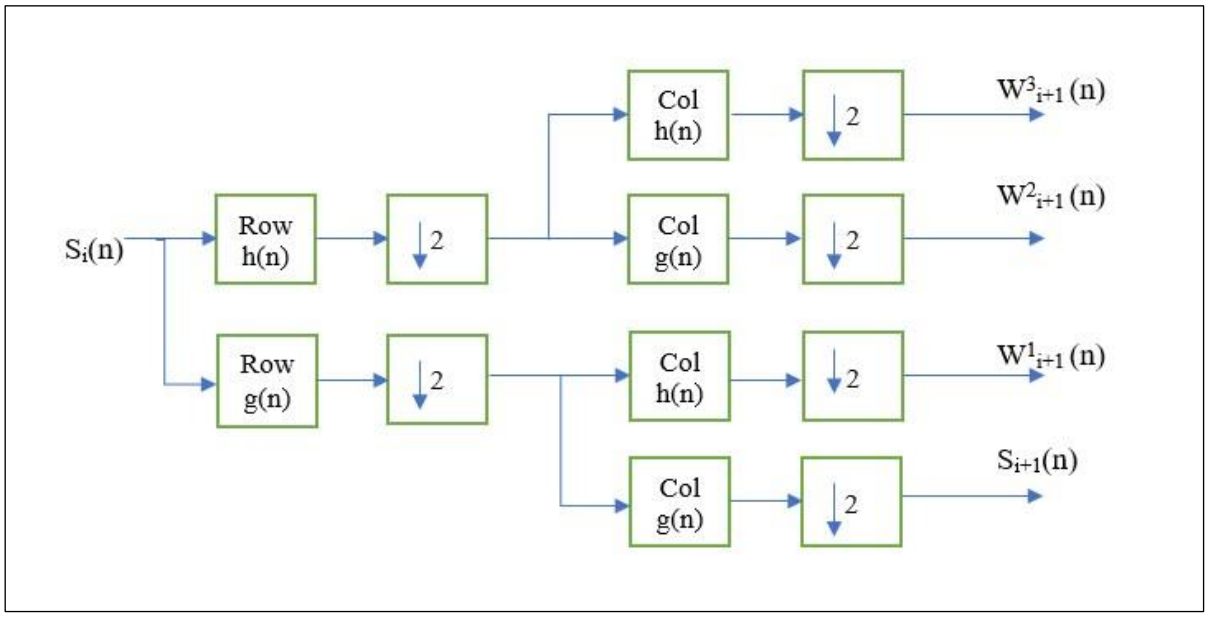

Fig 7: 2D-DWT

3.3 Hardware implementation of controller structure:

0	128	256	----------32640
1	129	257	----------32641
2	130	258	----------32642
3	131	259	----------32643
.	.	.	---------- .
.	.	.	---------- .
.	.	.	---------- .
.	.	.	---------- .
124	252	380	----------32764
125	253	381	----------32765
126	254	382	----------32766
127	255	383	----------32767

RAM only has enough for one dimension of data storage. The numbers in the above graph stand for RAM addresses. The column address is used while storing data in RAM. However, an address is now required whenever data is read from memory. The Fig 8 below depicts the three linked counters required to produce such addresses.

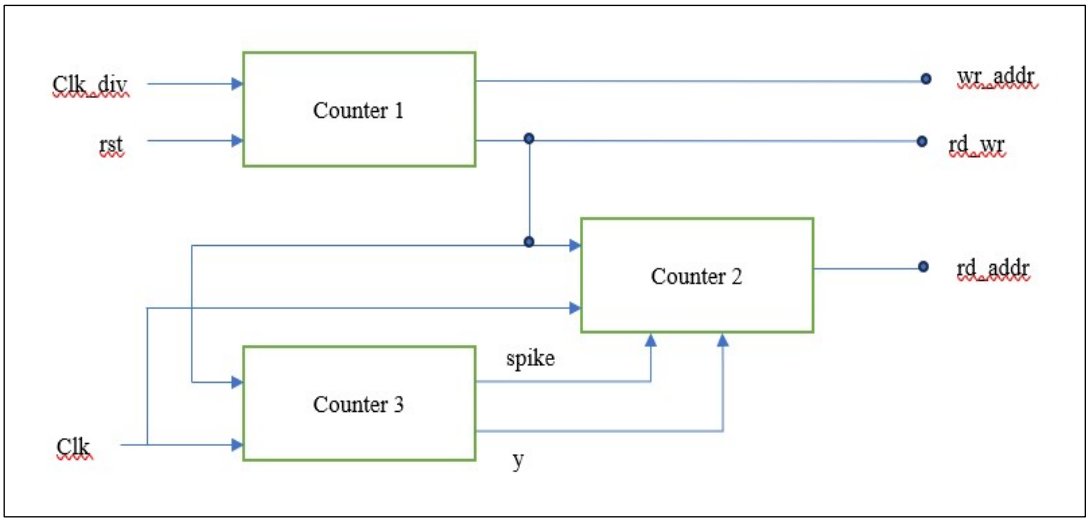

Fig 8: Controller Architecture Realized in Hardware

Counter 1:

Using "clk_ div" as clock 1, it counts up to 32367; at 32768, the wr _ address is frozen. To clarify, rd = wr = '1' under the current circumstances. As long as rd _ wr = '0' before the counter reaches 32768**.**

Counter 2:

The primary “clk” signal. This block's "rst" terminal connects to counter 1's "rd_wr" terminal. When rd_wr = 1, just this counter activates. Otherwise, off. So just this block activates when counter 1 hits 32768. Equation used:

$$y := y + '1';\ [\text{initial } y = '0']$$

If spike is 0, then y will continue at the same rate as before. The above 'x' equation should be carried out.

Counter 3:

This uses primary “clk” and “rd_wr” as “rst” like counter 2. It outputs “spike” and “y”. It utilizes equation:

$$a := a + '1'$$

when a = 128 make a = 0 & y = y + ‘1’
else y = previous y and run first ‘a’ equation.

3.4 The 2D-DWT Hardware Architecture

The Fig 9 the Hardware Structure of 2D-DW below depicts the fundamental hardware configuration of 2D-DWT. 1D-DWT, memory, a controller, mux and de-mux nodes round up the whole architecture. In the beginning, each signal is a 0. The '0' on the "Select" line causes the 0th terminal to be selected by the mux or de-mux. When this flag is set to 1, the first 1D-DWT block will become active and begin processing the picture data in serial form. Because of the de-mux setting, both the LP and HP data enter the memory block simultaneously at the clock signal specified by the 1st 1D-DWT block, which is simply clk/2. At that moment, the memory was in a writeable state (rd_wr = '0'). The rd_wr, rd_addr, and wr_address are supplied by the controller block. When the address in the write-protected region of memory (wr_addr) exceeds "32768," the controller switches the memory to reading mode (rd_wr = "1"). In the meanwhile, controller will produce rd_address. This block is only activated when rd_wr = '1', and its "LP" co-efficients are listed in the second 2D-DWT, whose "rst" is wired to the rd_wr line. Meanwhile, the "HP" block's information is being sent via the second Mux terminal. The de-mux's choose line is also wired to the rd_wr line. That's how the acquire of HL, HH, LH, LL pieces.After processing one picture frame (256 by 256 pixels), a separate counter is utilized to reset the whole block.

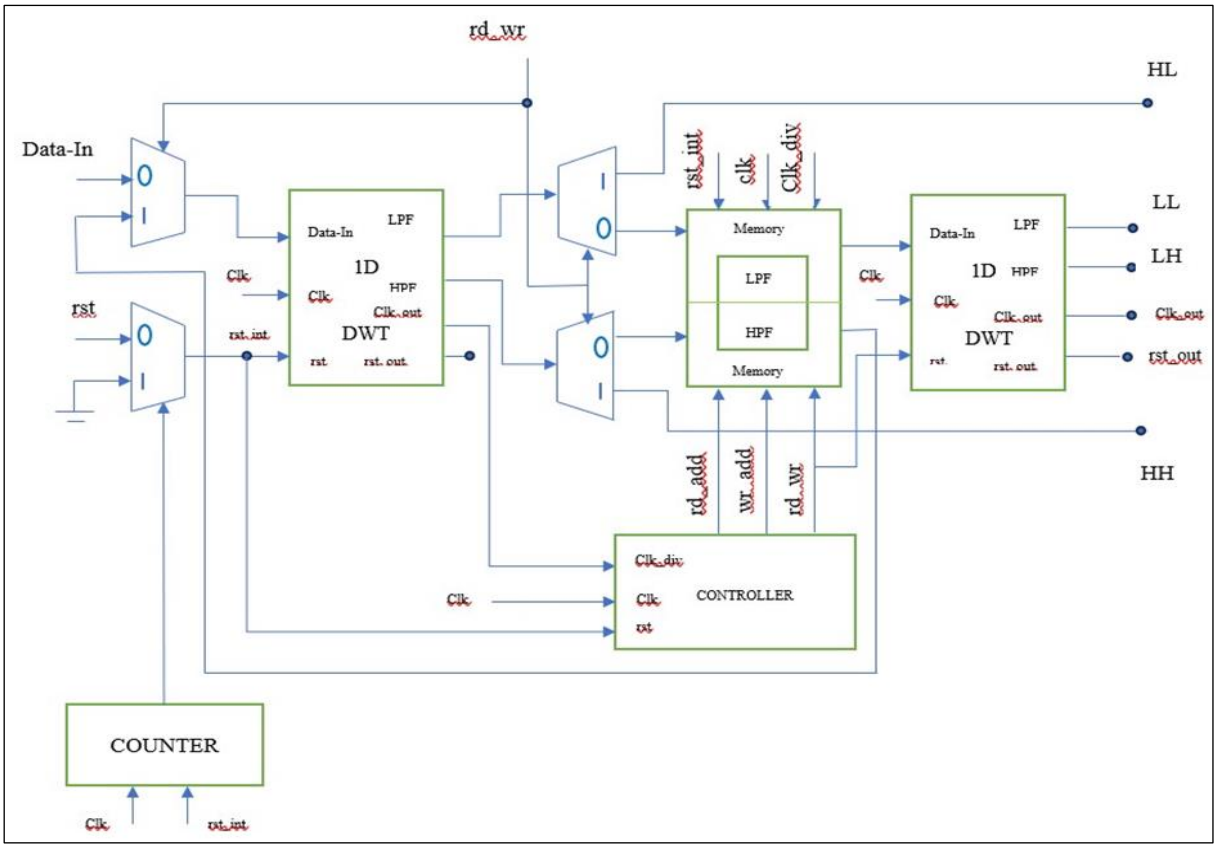

Fig 9: The 2D-DWT Hardware Architecture

For the 60th LPF and HPF side, there are pixels in the picture where the 1D-DWT before right shift yields a negative value. The difficulty in obtaining a '-ve' value is greater on the HPF side than on the LPF. It can ignore the negative numbers and treat them as zeros are know that the majority of an image's useful information is on the LPF side. "buffer-unit" is the term used for this function. This will check the most significant bit (MSB) to see whether the integer is negative. If the most significant bit is 1, the number is negative, and the buffer-unit's output is 0. In every other case, the number is positive and the output of buffer-unit is the actual number if the MSB unit is 0.

3.5 CDF-9/7 Using the CDF-5/3 Discrete Wavelet Transform

DWT Filter using a CDF-9/7 Basis: The Fig 10 below depicts the CDF-9/7's fundamental lifting design.

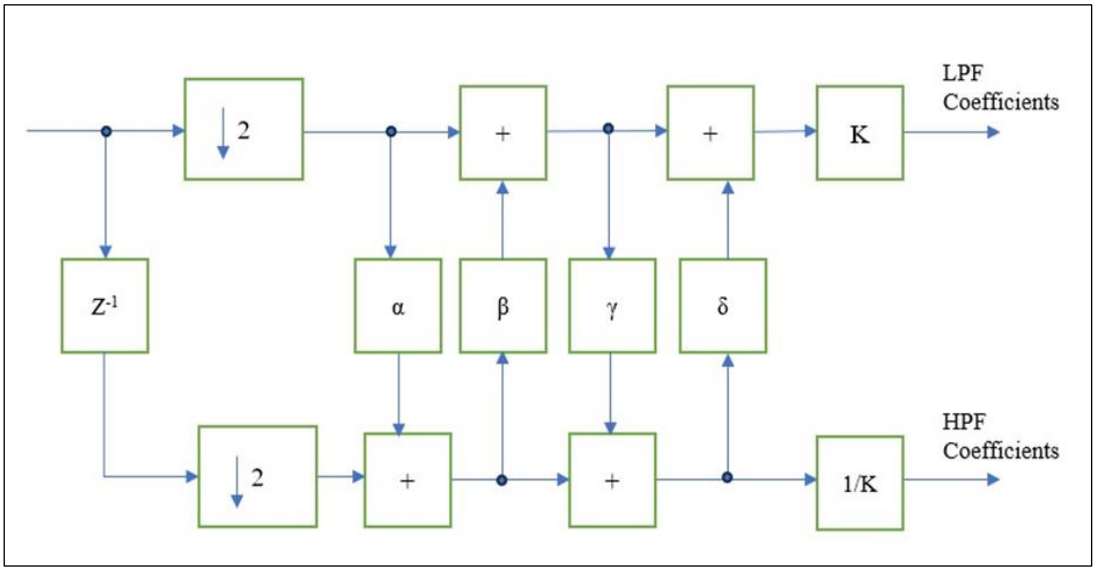

Fig 10: Input DWT Filter CDF-9/7 Base

The coefficients at the top of the picture are the lifting coefficients, whereas K is the scaling coefficient. All of the scaling and lifting factors have the values of

$\alpha = 1.586134342$
$\beta = 0.6529801185$
$\gamma = 0.882911076$
$\delta = -0.443506852$
$K = 1.149604398$
$1/K = 0.869864452$

All of the fractional coefficients, The Fractional parts are difficult to implement in hardware because they cause transaction errors and are inherently unstable.

This is why working on a CDF-5/3 DWT filter implementation, as this is the only filter type that requires just a shifter, and it should allow us to get the best possible transaction error performance.

4. Applications of FPGAS

Spartan-6 ATLYS (XC6SLX45-3CSG324) FPGA Board will be used throughout this chapter to explain the implementation of all recommended designs.

4.1 Top RTL Schematic

Fig 11 depicts the suggested 1D Top RTL schematic.

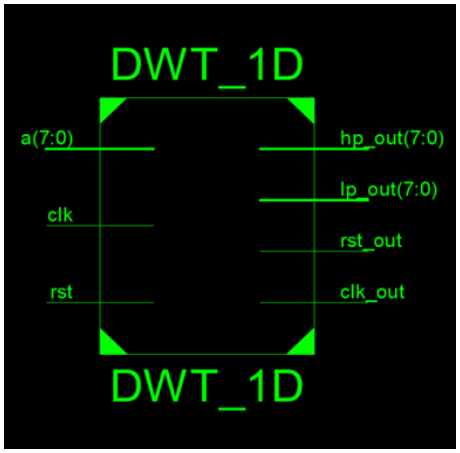

Fig 11: Top Level RTL Schematic of 1D-DWT 5/3 Block

4.2 Elaborated RTL Schematic

From top RTL schematic it is difficult to understand the internal components. So, Fig 12 shows the elaborated RTL schematic.

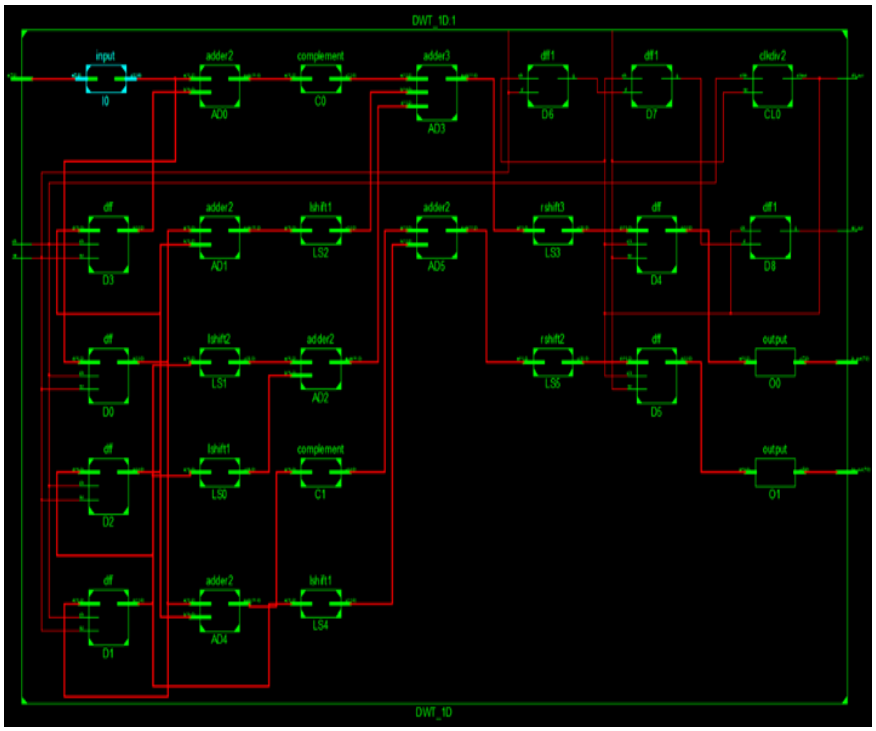

Fig 12: Elaborated RTL Schematic of 1D-DWT 5/3 Block

4.3 Technology Schematic

Each logic function will be represented by a set of LUTs and flip-flops in the FPGA. The technology diagram demonstrates the LUTs and flip-flops pairs used internally to achieve the suggested paradigm. In Fig 13 below, a technology plan for a 1D-DWT 5/3 block.

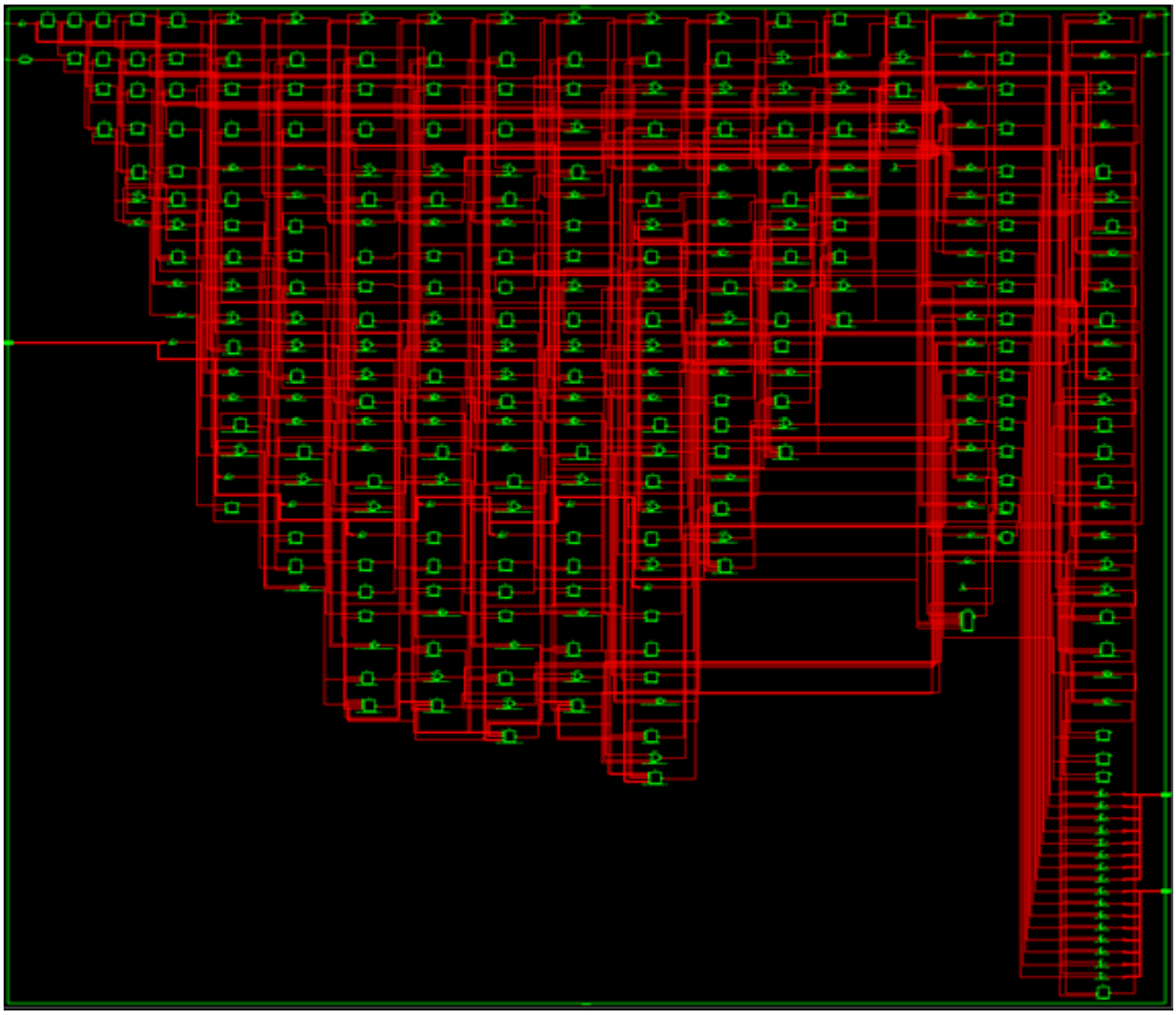

Fig 13: Technology Schematic of 1D-DWT 5/3 Block

4.4 Synthesis Result

Table 1 displays the results of the synthesis of the suggested CORDIC block.

Table 1: Synthesis Results of 1D-DWT 5/3

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	40	69120	0%
Number of Slice LUTs	107	69120	0%
Number of fully used LUT-FF pairs	18	129	13%
Number of bonded IOBs	28	640	4%
Number of BUFG/BUFGCTRLs	2	32	6%

4.5 Technology Schematic

Every logic function may be represented by a combination of LUTs and flip-flops in any FPGA. The technology diagram demonstrates how the suggested model is implemented internally using LUTs and flip-flops Fig 14 shows below.

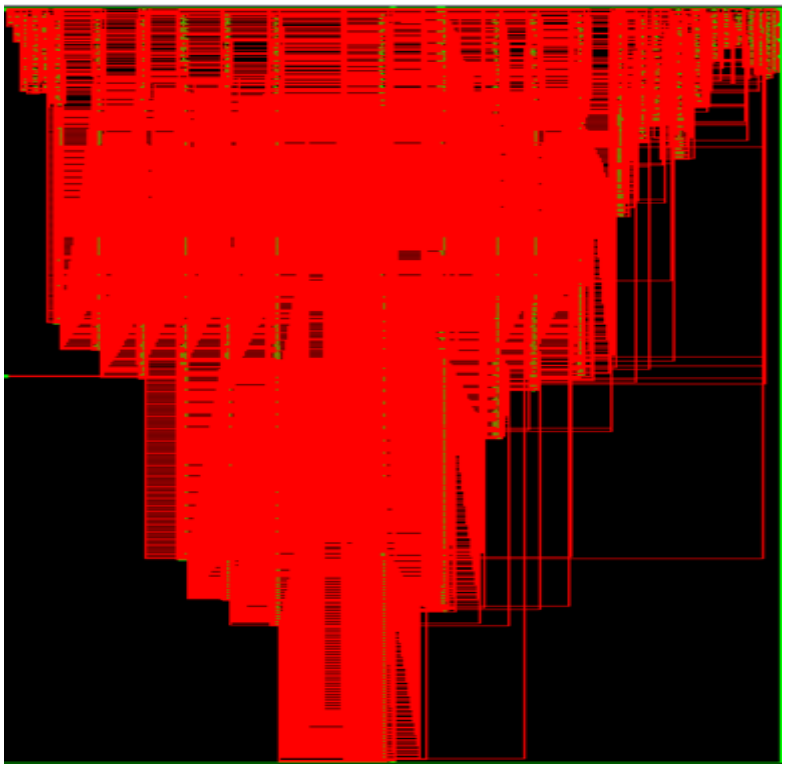

Fig 14: Technology Schematic of 2D DWT-9/7 Block

4.6 Synthesis Result

Table 2 displays the outcomes of the suggested block's synthesis.

Table 2: Analysis Results of 2D DWT-9/7 block

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	301	69120	0%
Number of Slice LUTs	9424	69120	13%
Number of fully used LUT-FF pairs	229	9496	2%
Number of bonded IOBs	44	640	6%
Number of BUFG/BUFGCTRLs	3	32	9%

5. RESULTS

- **Simulation Result of 1D DWT-5/3**

The simulation waveform of the proposed 1D DWT-5/3 block is shown in Fig 15.

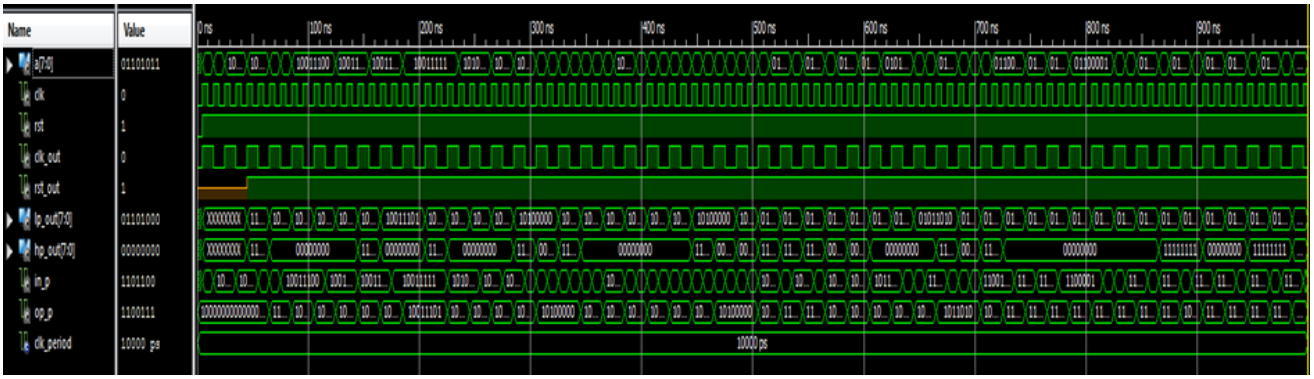

Fig 15: Simulation Waveform of CORDIC Block

- **Simulation Result of 2D DWT-5/3**

The simulation waveform of the proposed 2D DWT-5/3 is shown in Fig 16.

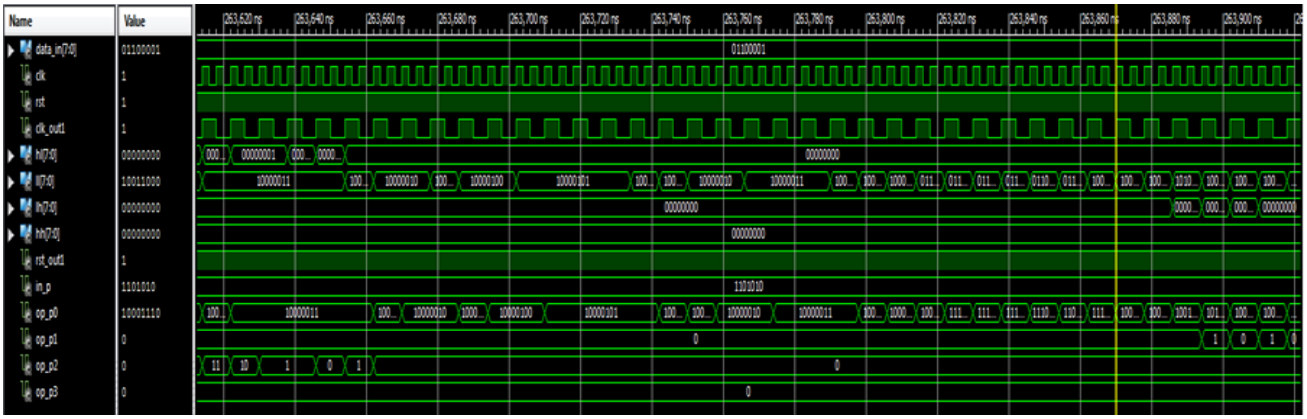

Fig 16: The Waveform of a 2D DWT-5/3 Block Simulation

◆ Simulation Result of Memory Architecture

The simulation waveform of the proposed block are shown in Fig 17.

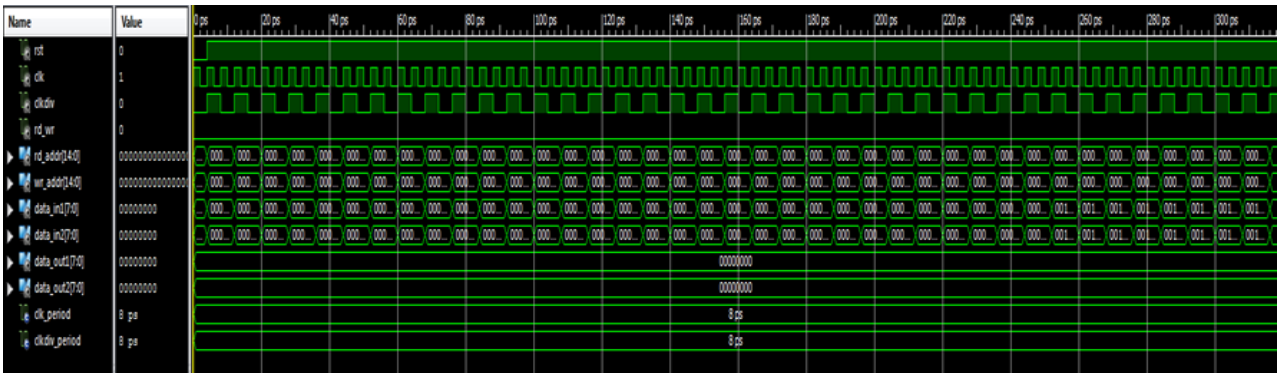

Fig 17: The Waveform of a 2D DWT-5/3 Block Simulation

◆ Simulation Result of Controller

The simulation waveform of the proposed block is shown in Fig 18.

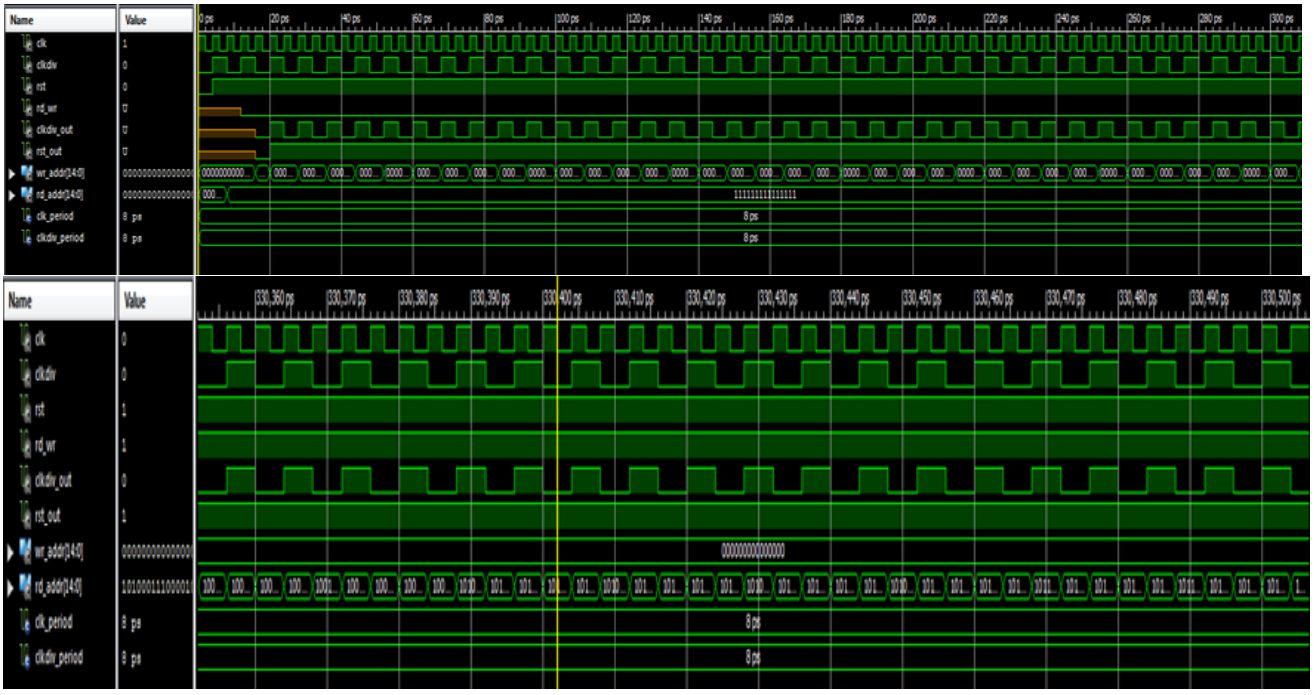

Fig 18: Simulation Waveform of Controller Block

◆ **Simulation Waveform of 1D DWT-9/7**

The simulation waveform of the proposed 1D DWT-9/7 block is shown in Fig 19.

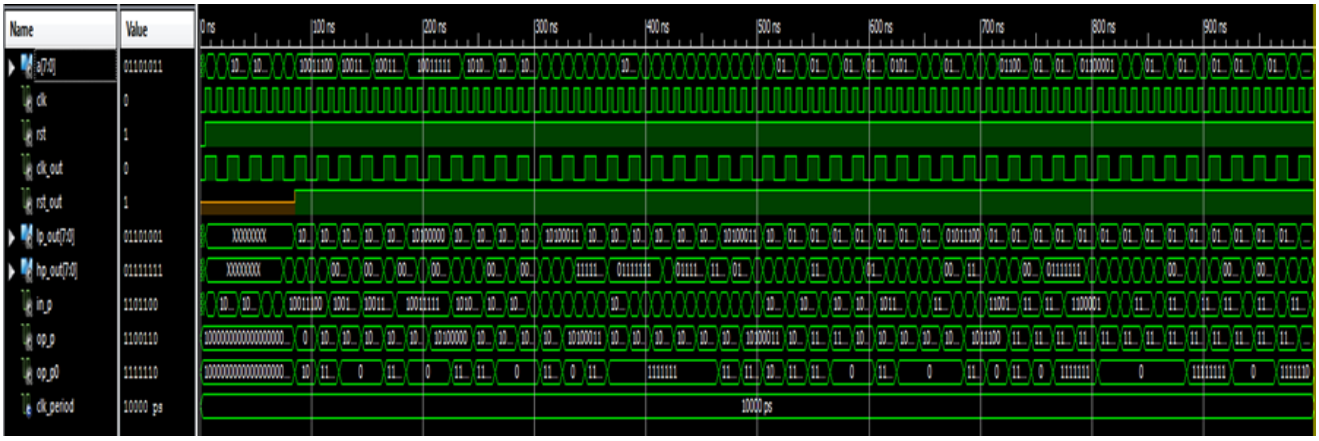

Fig 19: Simulation Waveform of CORDIC Block

◆ **Simulation Result of 2D DWT-9/7**

Fig 20 displays the simulated waveform of the proposed two-dimensional discrete wavelet transform (2D DWT-9/7).

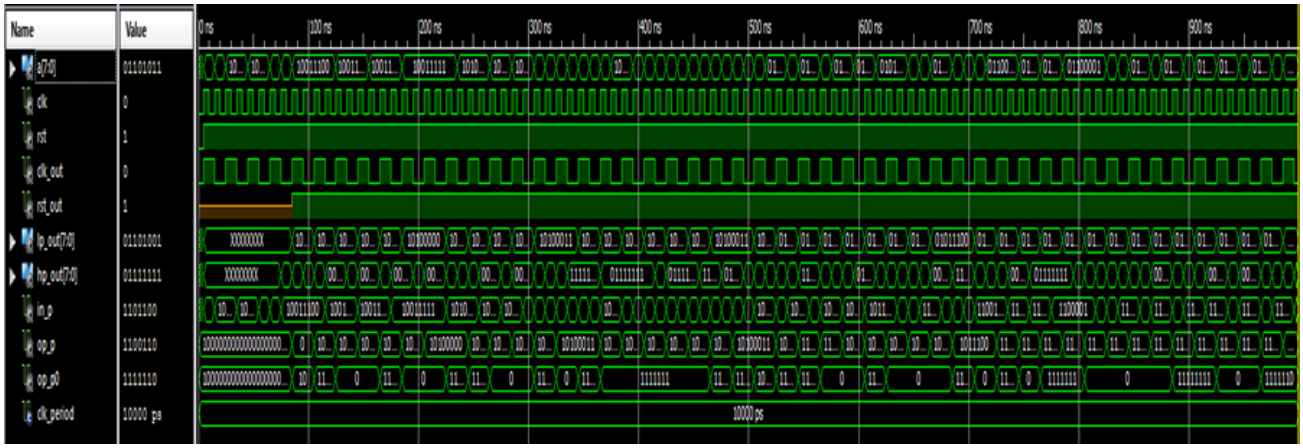

Fig 20: Simulation Wave of 2D DWT-9/7 Block

5.1 Comparison of Proposed Technique with Existing

This module provides a comparison of the proposed architecture to several already-existing ones.

5.2 2D DWT

In Table 3, how the proposed 2D DWT stacks up against the already-existing 2D DWT hardware. It can be seen from the table that the suggested 2D DWT design makes good use of the available hardware resources. The primary cause of this is the use of shifter modules in lieu of traditional multipliers and divisions for calculations requiring constant inputs.

Table 3: Comparisons of Existing 2D DWT with Proposed 2D DWT

Parameters	Yasodai and Ramaprasad	Rudagi and Vinayak	Proposed
Board	Virtex-4	Virtex-6	Spartan-6
Number of Slices	520	965	475
Number of Slice Flip Flops	936	----	418

6. CONCLUSION

In this work, a new approach to memory-efficient DWT architecture implementation. Furthermore, the suggested design may be scaled up to a considerably larger number of DWTs without significantly altering the twiddle factor generators. In addition to cutting down on hardware needs, the butterfly diagram's use of intermediate phases allows more precise processing with fewer components. A face detection utilizing the suggested DWT architecture, where the proposed DWT is utilized to extract database and test features, to demonstrate the practical uses of the

proposed architecture. In the future, to optimize data precision and hardware parameters and update the DWT architecture to support more butterflies.

REFERENCE

- Suresh, K. D., Vijaya, B. B., & Suryaprakash, R. (2012). Implementation of Higher Order DWT Processor Using FPGA, *IJERT*, *1*(6), 1-4.
- Aniket, S., Mayuresh, D. (2012). Comparative Study of Various DWT Algorithm Implementation on FPGA, *IJETS*, *1*(1), 19-22.
- Gupta, A., Jain, A., Bhalla, A. V., & Malviya, U. (2012). Design Of High Speed FFT Processor Using Vedic Multiplication Technique. *International Journal of Engineering Research and Applications (IJERA)*, *2*(5), 1501-1504.
- Sri, T. S., Kumar, K. S., & Muttaiah, R. (2013). Review of CORDIC Architectures. *International Journal of Engineering and Technology*, *5*(2), 578-585.
- Narayanam, R., & Muni, G. P. (2013) Performance Evaluations of Grigoryan DWT and Cooley-Tukey DWT onto Xilinx Virex-II pro and Virtex-5 FPGA, *International Journal of Scientific and Research Publications*, *3*(1), 1-6.
- Sneha, K. (2015). Design and Implementation of DWT, *International Journal of Electronics, Communicatin and Soft Computing Science and Engineering*, pp. 248-252.
- Patrikar, M., & Tehre, V. (2017). Design and Power Measurement of 2 and 8 Point FFT using Radix-2 Algorithm for FPGA Implementation. *IOSR Journal of VLSI and Signal Processing*, *7*(1), 44-48.
- Amaresh, K., Tripathi, U. N., Roopak, K. V., & Manish, M. (2014). 64-Point Radix-4 DWT Butterfly Realization using FPGA, *International Journal of Engineering and Innovative Technology*, *4*(4), 57-60.
- Gitla, S., Masthaniah, P., Veeranth, P., & Durga, G. R. (2013). VLSI Implementation of a Flexible and Synthesizable DWT Processor, *IRD India*, *1*(3), 73-77.
- Yasodai, A., & Ramprasad, A. V. (2013). A New Memory Reduced Radix-4 CORDIC Processor for DWT Operation, *IOSR Journal of VLSI and Signal Processing*, *2*(5), 9-16.
- Rudagi, J. M., & Vinayak, D. (2013). Radix-2 CORDIC Method with Constant Scale Factor, *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, *2*(7), 3408-3413.
- Arman, C., Yousuef, S. K., Otto, S., & Radha, R. (2011). Implementing DWT Algorithm on FPGA, *International Journal of Computer Science and network Security*, *11*(11), 148-156.
- Akashadip, A. J., Prashant, R. I., & Ravindra, D. K. (2017). Design and Simulation of Floating Point DWT Processor based on Radix-4 Algorithm using VHDL, *International Journal of innovative Research in Computer and Communication Engineering*, *4*(7), pp. 13223-13229.
- Patrikar, M., & Tehre, V. (2017). Design and Power Measurement of 2 and 8 Point FFT using Radix-2 Algorithm for FPGA Implementation. *IOSR Journal of VLSI and Signal Processing*, *7*(1), 44-48.
- Anilkumar, P. H., & Beulet, P. A. S. (2015). Lifting-based discrete wavelet transform for real-time signal detection. *Indian Journal of science and technology*, *8*(25), 1-6.