VIDEO COMPRESSION USING LIFTING BASED DWT ALGORITHM AND IMPLEMENTATION ON FPGA

S.Thavasipitchiahraja, Student
M. Tech - VLSI & Embedded System
B.S.Abdur Rahman University
Chennai, India
Pitchiah77@gmail.com

Mrs.V.Kavitha, Assistant Professor
Department Of ECE
B.S.Abdur Rahman University
Chennai, India
vkavitha@bsauniv.ac.in

***Abstract* —In this paper, we tend to review recent developments in VLSI architectures and algorithms for implementation of lifting, which is primarily based on Discrete Wavelet Transform (DWT). The fundamental principle behind the lifting based algorithm is to decompose the finite impulse response (FIR) filters into a finite sequence of easy filtering steps. Lifting is primarily based on DWT implementation, which has several benefits and recently been projected for the JPEG2000 compression. Consequently, in recent years this has become an active analysis in number of other architectures. During this paper, we offer a survey of those architectures for 2-dimensional DWT and lifting 2D-DWT was implemented on Spartan-3E FPGA using System C Coding.**

***Index Terms*—2-DWT;VHDL;Microblaze;EDKTool, FPGA**

I. INTRODUCTION

Over last decade the discrete wavelet transform (DWT) has became a power full signal processing tool. Since mat-lab planned the multi-resolution of signals, the discrete wavelet transform has been effectively employed in signal and image processing applications. The advantage of DWT over alternative ancient transformations is that, it performs multi-resolution analysis of signals in both time and frequency. Currently for compression technique DWT has been used widely as it supports operation like image transmission (by quality, by resolution) and easy compressed image manipulation. In fact, compared to the present JPEG standard, It is used in the new JPEG2000 [1] standard for superior performance of compression.

Historically DWT has been enforced by convolution FIR filter bank structures. Such FIR filter bank structure implementations require an outsized variety of arithmetic computations and an outsized storage features. The area unit is for either high speed or low power image/video process applications. A new mathematical formulation for moving ridge transformation has been projected by Sweden [2], for the support of abstraction construction of the wavelets and an awfully versatile theme for its factorization [3].

This new Approach in DWT is termed as the lifting-based moving ridge re-work or simply lifting. The important feature of the lifting-based DWT theme is to interrupt with the high-pass and low-pass wavelet filters into a sequence of higher and lower triangular matrices and then it is also used to convert the filter implementation into banded matrix multiplications [4].

This matrix multiplications often needs fewer computations compared to the convolution based DWT [5], which also offers several different advantages. The most important feature of lifting-based DWT has been triggered in the development of architecture in recent years.

These architectures vary from extremely parallel architectures, to programmable DSP-based architectures. We tend to present a survey of those architectures in this paper and we also offer a systematic derivation of those architectures and comment on their hardware and timing requirements.

II. BLOCK DIAGRAM

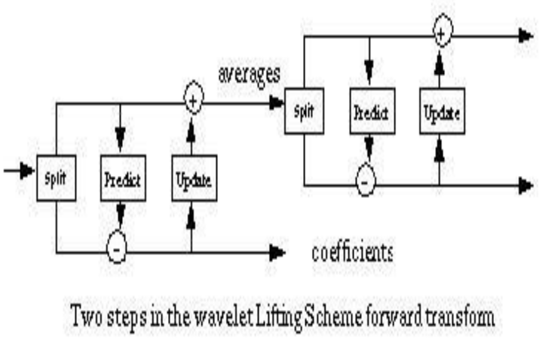


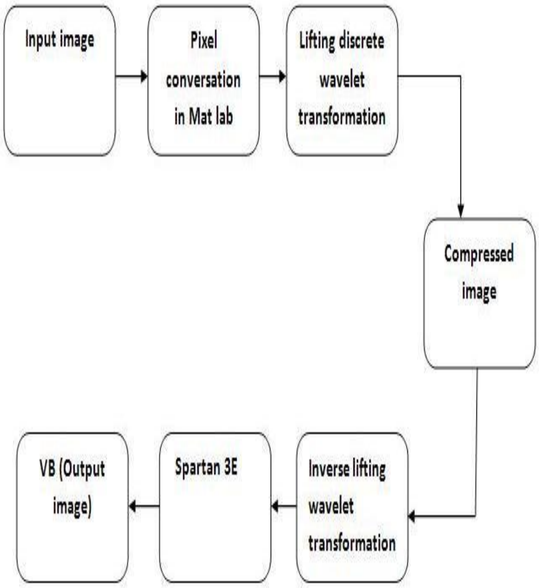
FIG. 1: LIFTING SCHEME PROCESS FLOW

It is composed of three basic operation stages:

- Splitting: where the signal is split into even and odd[3]

- Predicting: Even samples are multiplied by a predict factor[3]
- Updation: The detailed coefficients computed by the predict step are multiplied by the update factors and then the results are added to the even samples to get the coarse coefficients.[3]

III. IMPLEMENTATION PROCESS


Input image
Pixel conversation in Mat lab
Lifting discrete wavelet transformation
Compressed image
Inverse lifting wavelet transformation
Spartan 3E
VB (Output image)

IV. WAVELET TRANSFORM

Mathematical functions of wavelets are defined over a finite interval, having an average value of zero, which is used to transform data into different frequency components, representing each component with a resolution of the scale matched.

The basic idea of the wavelet transform is to represent any arbitrary function into a superposition of its wavelet set or into a basic function. These basic functions are called as baby wavelets which are obtained from a single prototype wavelet called the mother wavelet, by the dilations of contractions (scaling) and translations (shifts).

They also have advantages over traditional Fourier methods, by analyzing physical situations; the signal contains discontinuities and sharp spikes. Many new wavelet applications are developed in recent years, such as image compression, turbulence, human vision, radar, and earthquake prediction.

In wavelet transform the basic functions are wavelets. They tend to be irregular and symmetric. All wavelet functions, such as *w* (2*kt* - *m*), are derived from a single mother wavelet, *w* (*t*).As shown in (Fig2)

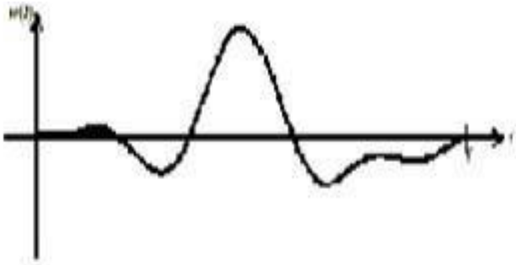


FIG. 2: MOTHER WAVELET W(T)

Normally it starts at time *t* = 0 and ends at *t* = *T*. The shifted wavelet *w* (*t* - *m*) starts at *t* = *m* and ends at *t* = *m* + *T*. The scaled wavelets *w* (2*kt*) start at *t* = 0 and end at *t* = *T*/2*k*. Their graphs are *w* (*t*) compressed by the factor of 2*k* as shown in Fig. 3. For example, when *k* = 1, the wavelet is shown in Fig 3 (a). If *k* = 2 and 3, they are shown in (b) and (c), respectively.

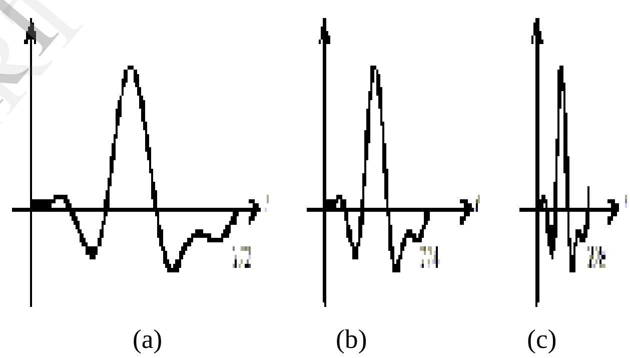

(a) (b) (c)

FIG. 3: SCALED WAVELETS

These wavelets are called orthogonal wavelets and their inner products are zero. The smaller the scaling factor is, the wider the wavelet. Wide wavelets are comparable for low-frequency sinusoids and narrow wavelets are comparable for high-frequency sinusoids.

V. 2-D TRANSFORM HEIRARCHY

The 1-D wavelet transform can be extended to a two-dimensional wavelet transform using separable wavelet filters. With the use of separable filters, the 2-D wavelet transform can be computed by applying a 1-D wavelet transform to all the rows and then repeating for all the columns.

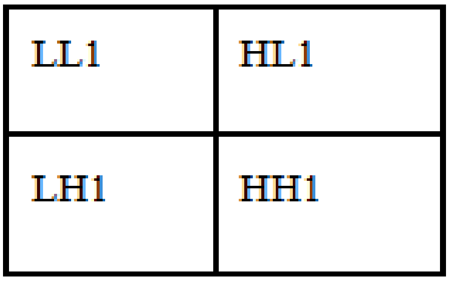


FIG. 4: SUB-BAND LABELING SCHEME FOR A ONE LEVEL, 2-D WAVELET TRANSFORM

The original image of one-level (K=1), 2-D wavelet transform, with corresponding notation is shown in Fig. 4. This example is repeated for a three-level (K =3) wavelet expansion in Fig. 5. The highest level of the decomposition of the wavelet transform is represented by the k value.

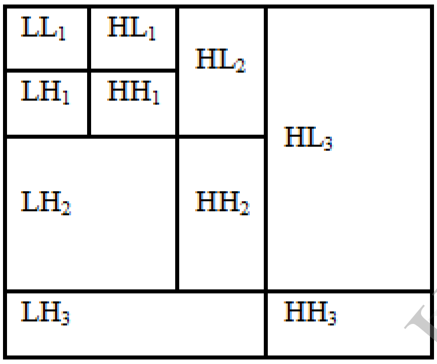


FIG. 5: SUB-BAND LABELING SCHEME FOR A THREE LEVEL, 2-D WAVELET TRANSFORM

The extension of 1-D sub-band decomposition is 2-D sub-band decomposition. By executing 1-D sub-band decomposition twice, the entire process is carried out as first in one direction (horizontal) and then in the orthogonal (vertical) direction. The low-pass sub-bands (Li) resulting from the horizontal direction is further decomposed into vertical direction, leading to LLi and LHi sub-bands. Similarly, the high pass sub-band (Hi) is further decomposed into HLi and HHi are the examples of 2-D transform.

After one-level of transform, the image can be further decomposed by using the 2-D sub-band decomposition to the existing LLi sub-band. This iterative process results in multiple “transform levels”. In Fig.4 the first level of decomposition results are shown as LH1, HL1, and HH1, further to LL1, which is decomposed into LH2, HL2, HH2, LL2 at the second level decomposition, and the information of LL2 is used for the third level decomposition in fig.5. The sub-band LLi is a low-resolution and high-pass sub-bands LHi, HLi, HHi are horizontal, vertical, and diagonal respectively. Since, they represent the horizontal, vertical, and diagonal residual information of the original image.

VI. DWT LIFTING IMPLEMENTATION

To speed up the process parallel implementation of the Distributive Arithmetic (DA) architecture shown in Fig 6 is realized in. In parallel implementation, the input data is divided into even samples and the odd samples based on their position. This scheme reduces the memory size to half due to the symmetric property of the filter coefficients. This increases the through put as the input samples are simultaneously used to read the data from two LUTs and hence speed is increased.

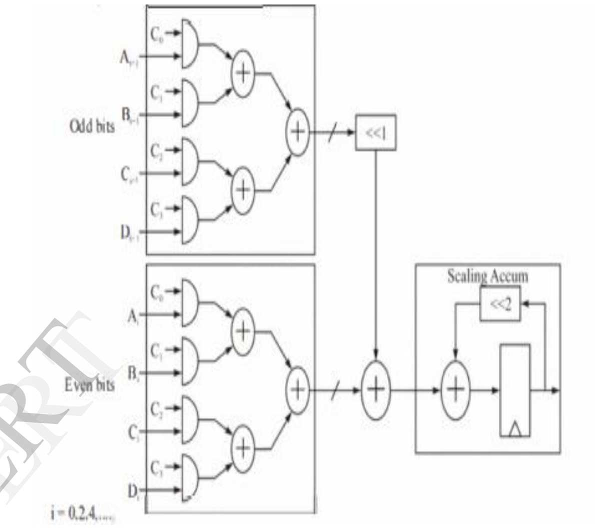


FIG. 6: INTERNAL BLOCK DIAGRAM OF LIFTING DWT

The modified DA-DWT architecture shown in Fig 7 consists of four LUTs, each of the LUTs are accessed by the even and odd samples of input matrix simultaneously. Odd and even input samples are divided into 4 bits of LSB and 4 Bits of MSB, each 4-bit data read the content of four different LUTs that consist of partial products of filter values computed and stored as per the DA logic. Input samples are split into even and odd in the first stage, the data is further loaded sequentially into the serial in serial out shift registers, top four shift register store MSB bits and bottom four shifts register stores the LSB bits. It requires 40 clocks cycles to load the shift register contents. At the end of 40th clock cycle, the control logic configures the shift register as serial in parallel out, thus forming the address for the LUT. The partial products stored in the LUT are read simultaneously front all the four LUTS and are accumulated with previous values Available across the shift register in the output stage. The output stage consisting of adders, accumulators and right shift registers are used to accumulate the LUT contents and thus compute the DWT output. This architecture has a latency of 44 clock cycles in computing the first high pass and low pass filter coefficients, and has a through put of 4 clock cycles. This architecture is faster by the previous architectures as the latency is reduced by half clock cycles and through put is increased by a factor of 2.

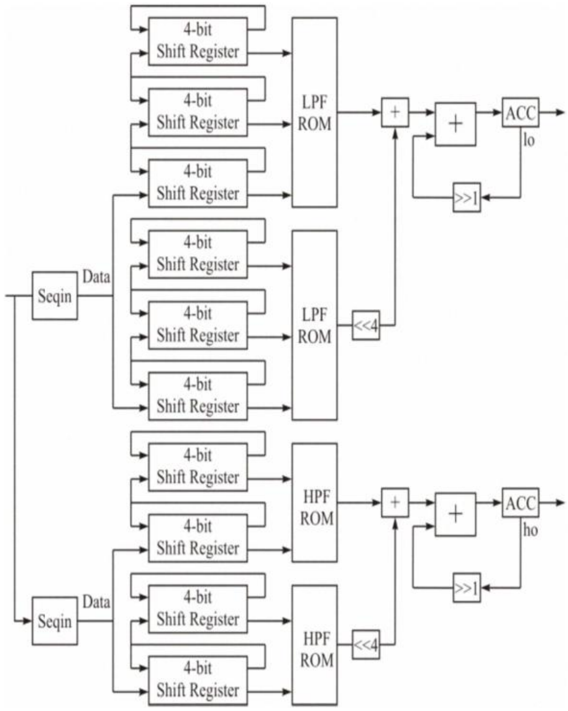


FIG. 7: BLOCK DIAGRAM OF PARALLEL LIFTING DWT

THE BASIC DA EQUATION CAN BE GIVEN

$$x_n = -x_{n,l}.2^l + \sum_{b=0}^{l-1} x_{n,b}.2^b \tag{1}$$

where l = (total number of bits per sample). In dyadic space, a convolution-based wavelet filter can be represented as,

$$w_a = \sum_n x_n h_{2a-n} \tag{2}$$

where *xn* and *hn* are input samples and filter coefficients, respectively. Considering frame-length = 4 and wordlength = 4 (as an example) and using (1) in (2) with *a* = 1, we get

$$w_1 = -[x_{2,3}h_0 + x_{1,3}h_1 + x_{0,3}h_2]2^3 + \cdots$$
$$+[x_{2,0}h_0 + x_{1,0}h_1 + x_{0,0}h_2]2^0$$
3)

where *xij* is the *i*th sample’s *j*th bit of the input data.

The possible combinations of filter coefficients obtained from are shown in the first six rows, which occupies 16 memory locations. However, it can be observed that there exists redundant (such as “0”) and repetitive filter coefficients (such as “*h*0,” “*h*1,” “*h*2,” “*h*1 + *h*2,” “*h*0 + *h*1,” “*h*0 + *h*2,” and “*h*0 + *h*1 + *h*2”) occupying more than a single memory location. Thus, if only the unique combinations of the filter coefficients are stored in the memory, the other filter coefficients can be obtained on the fly using simple addition operations. In this particular example, the proposed methodology leads to only four memory locations, as shown in the last two rows, rather than 16 locations in the conventional approach.

VII. EXPERIMENTAL SETUP

A. Xilinx Platform Studio:

For designing the hardware portion of our embedded processor system, the Xilinx platform studio (XPS) is used as the development environment or GUI.

B. Embedded Development Kit:

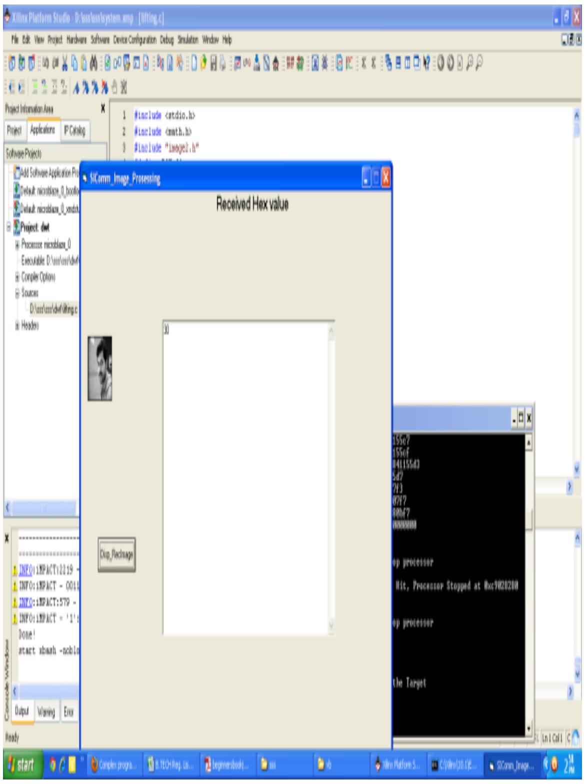
For developing embedded systems with Xilinx Micro-Blaze and Power-PC CPUs, the Xilinx Embedded Development Kit (EDK) is used as an integrated software tool. To develop an embedded system, the EDK includes a variety of tools and applications for the assistance of the designer and it helps in the hardware creation to final implementation of the system on a FPGA.

The creation of the hardware and software components of embedded system is present in system design. The verification of component is optional in embedded system. A basic embedded system design project involves: hardware platform creation, hardware platform verification (simulation), software platform creation, software application creation and software verification.

C. Software Development Kit

Xilinx Platform Studio Software Development Kit (SDK) is an integrated development environment which is complimentary to XPS which is used for C or C++ embedded software application creation and verification. SDK is built on the Eclipse which is used as an open source framework. Soft Development Kit (SDK) is a suite of tools which is used to enable you to design a software application for selected Soft IP Cores in the Embedded Development Kit (EDK).The software application can be written in a C or C++ language. Then the complete embedded processor system is used for user application which will be completed by doing debug and downloading the bit file into FPGA. Then FPGA behaves like processor which is implemented in a Xilinx Field Programmable Gate Array (FPGA) device.

VIII. RESULTS

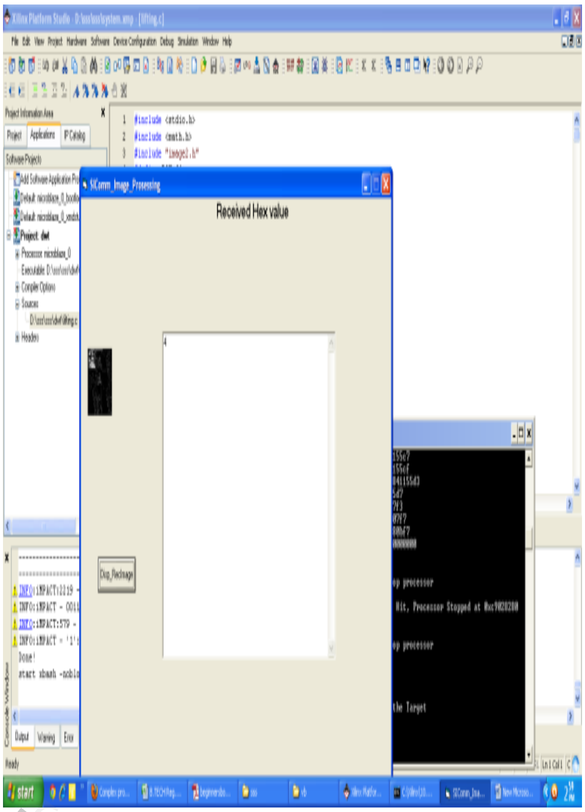


```
#include <stdio.h>
#include <math.h>
#include "image1.h"
```

```
INFO:iMPACT:2219 -
INFO:iMPACT - 001
INFO:iMPACT:579 -
INFO:iMPACT - '1':
Done!
start xbash -nobl
```

```
155c7
155cf
041155d3
5d7
7f3
07f7
00bf7
0000000

op processor

 Hit, Processor Stopped at 0xc9020200

op processor

the Target
```

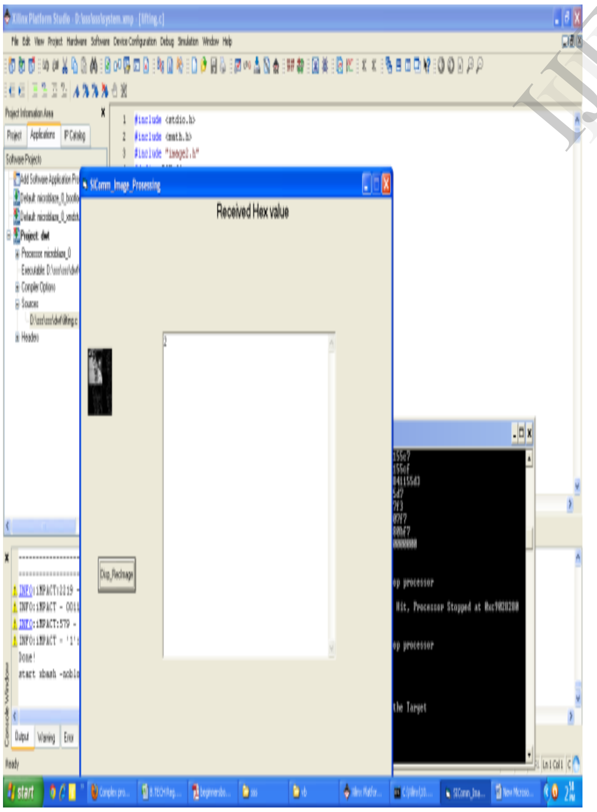
FIG.8: INPUT IMAGE READ THROUGH VB SCREEN



```
#include <stdio.h>
#include <math.h>
#include "image1.h"
```

```
155c7
155cf
041155d3
5d7
7f3
07f7
00bf7
0000000

op processor

 Hit, Processor Stopped at 0xc9020200

op processor

the Target
```

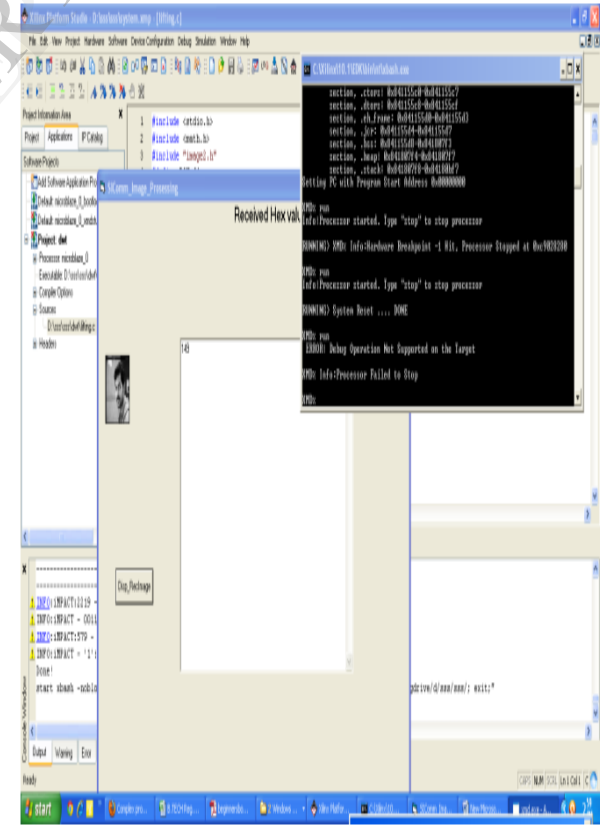
FIG.10: 3 LEVEL 2D-DWT IMAGE



```
#include <stdio.h>
#include <math.h>
#include "image1.h"
```

```
155c7
155cf
041155d3
5d7
7f3
07f7
00bf7
0000000

op processor

 Hit, Processor Stopped at 0xc9020200

op processor

the Target
```

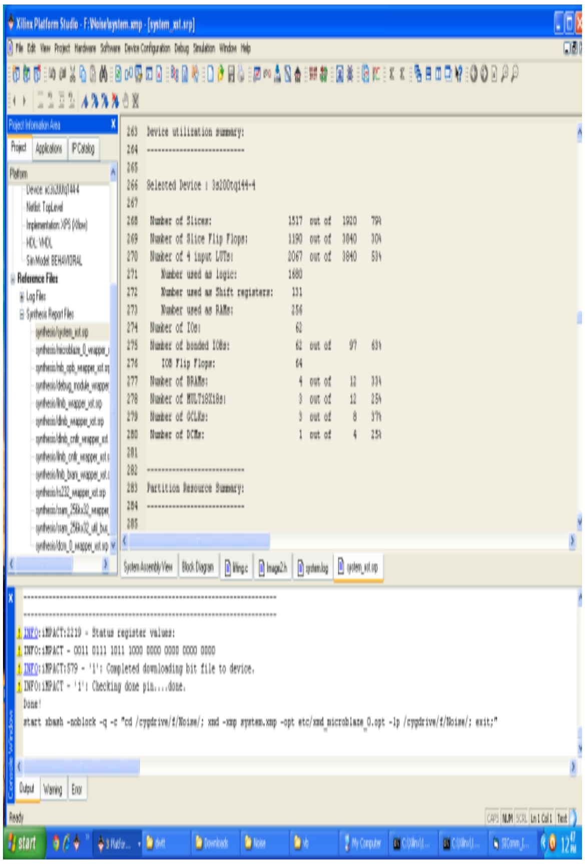
FIG.9: DWT IMAGE



```
#include <stdio.h>
#include <math.h>
#include "image1.h"
```

```
C:\Xilinx\10.1\EDK\bin\nt\xbash.exe
        section, .ctors: 0x841155c0-0x841155c7
        section, .dtors: 0x841155c8-0x841155cf
        section, .eh_frame: 0x841155d0-0x841155d3
        section, .jcr: 0x841155d4-0x841155d7
        section, .bss: 0x841155d8-0x841007f3
        section, .heap: 0x841007f4-0x841007f7
        section, .stack: 0x841007f8-0x841008f7
Setting PC with Program Start Address 0x00000000

XMD% run
Info:Processor started. Type "stop" to stop processor

RUNNING> XMD% Info:Hardware Breakpoint -1 Hit, Processor Stopped at 0xc9020200

XMD% run
Info:Processor started. Type "stop" to stop processor

RUNNING> System Reset .... DONE

XMD% run
 ERROR: Debug Operation Not Supported on the Target

XMD% Info:Processor Failed to Stop

XMD%
```

FIG.11: ORIGINAL RECONSTRUCTED AFTER INVERSE DWT


Xilinx Platform Studio - F:\Noise\system.xmp - [system_xst.srp]
File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help
System Assembly View
Block Diagram
lifting.c
Image2.h
system.log
system_xst.srp
Output
Warning
Error
Ready

```
Device utilization summary:
---------------------------

Selected Device : 3s200tq144-4

 Number of Slices:                    1517  out of   1920    79%
 Number of Slice Flip Flops:          1190  out of   3840    30%
 Number of 4 input LUTs:              2067  out of   3840    53%
    Number used as logic:             1680
    Number used as Shift registers:    131
    Number used as RAMs:               256
 Number of IOs:                         62
 Number of bonded IOBs:                 62  out of     97    63%
    IOB Flip Flops:                     64
 Number of BRAMs:                        4  out of     12    33%
 Number of MULT18X18s:                   3  out of     12    25%
 Number of GCLKs:                        3  out of      8    37%
 Number of DCMs:                         1  out of      4    25%

---------------------------
Partition Resource Summary:
---------------------------
```

```
--------------------------------------------------------------------
--------------------------------------------------------------------
INFO:iMPACT:2219 - Status register values:
INFO:iMPACT - 0011 0111 1011 1000 0000 0000 0000 0000
INFO:iMPACT:579 - '1': Completed downloading bit file to device.
INFO:iMPACT - '1': Checking done pin....done.
Done!
start xbash -noblock -q -c "cd /cygdrive/f/Noise/; xmd -xmp system.xmp -opt etc/xmd_microblaze_0.opt -ip /cygdrive/f/Noise/; exit;"
```

FIG.12: SYNTHESIS REPORT

IX. FUTURE SCOPE

Memory Efficient VLSI Modular architecture can be designed Using 3-D DWT based on lifting method. This gives memory efficient and hi-speed VLSI modular architecture throughput. This is also used to reduce the output time.

X. CONCLUSION

This paper presented an approach towards VLSI implementation of the lifting based Discrete Wavelet Transform (DWT) for video to frames, compression. Lifting based DWT is used for the implementation and has many advantages and also been recently proposed for the JPEG2000 standard of image compression. Consequently, in recent years this has become an area of active research on several architectures which have been proposed. In this paper, we provide architecture for 2-dimensional lifting DWT. The Discrete Wavelet Transform provides a multi resolution representation of images. The transform has been implemented using filter banks. The design based on the constraints, the area, power and timing performance were obtained. Modified DA technique was implemented. The latency of the proposed architecture is 44 clock cycles and throughput is 4 clock cycles, and hence is twice faster than the reference design.

XI. ACKNOWLEDGMENT

I would like to thank my project supervisor Mrs. V. Kavitha, who without her help and guidance this project would not have completed and would also like to HOD, ECE Department and other Professors for extending their help & support in giving technical ideas about the paper and motivating to complete the work effectively & successfully.

REFERENCES

[1] Wei Zhang,ZheJiang,ZhiyuGao, and YanyanLiu,"An efficient VLSI architecture for Lifting based discrete wavelet transform,"IEEETrans.Circuits and systems,vol.59,NO.3,pp. 158-162,Mar.2012.

[2] G. Xing, J. Li, and Y. Q. Zhang, "Arbitrarily shaped videoobject coding by wavelet," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 10,pp. 1135–1139, Oct. 2001.

[3] S. C. B. Lo, H. Li, and M. T. Freedman, "Optimization of wavelet decomposition for image compression and feature preservation," IEEE Trans.Med.Imag., vol. 22, no. 9, pp. 1141–1151, Sep. 2003.

[4] K. K. Parhi and T. Nishitani, "VLSI architecture for discrete wavelet transforms," IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 1, no. 2, pp. 191–202, Jun. 1993.

[5] P.Wu and L. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 4, pp. 536–545, Apr. 2001.

[6] W. Sweldens, "The new philosophy in biorthogonal wavelet constructions, "in Proc. SPIE., 1995, vol. 2569, pp. 68–79.

[7] Daubechies and W. Sweldens, "Factoring wavelet transform into lifting steps," J. Fourier Anal. Appl., vol. 4, no. 3, pp. 245–267, Mar. 1990

[8] T. Acharya and A. K. Ray, Image Processing: Principles and Applications. Hoboken, NJ: John Wiley & Sons, 2005

[9] Chin-Fa Hsieh, Tsung-Han Tsai and Chih-Huang Lai, "Implementation of an efficient DWT using a FPGA on a Real-time Platform," IEEE, ICICIC, Second International Conference on, pp. 235-235, 2007

[10] P.Y Chen, "VLSI implementation for one-dimensional multilevel lifting based wavelet transform ," IEEE Trans. on Computers, vol. 53, pp.386- 398, 2004.

[11] XuguangLan, Nanning Zheng and Yuehu Liu, "Low-power and high-speed VLSI architecture for lifting-based forward and inverse wavelet transform," IEEE Trans. on Consumer Electronics, Vol.51, pp.379-385, 2005.

[12] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Flippingstructure:An efficient VLSI architecture for lifting based discretewavelet transform,"IEEE Trans. Signal Process., vol. 52, no.4,pp. 1080–1089, Apr. 2004.