IOSR Journal of Electronics and Communication Engineering (IOSRJECE)
ISSN : 2278-2834 Volume 2, Issue 5 (Sep-Oct 2012), PP 18-25
www.iosrjournals.org

An FPGA Based Microblaze Soft Core Architecture for 2D-LIFTING

[1]K. Priyanka, [2]M. Pradeep,
[1](*VLSI System design, ECE Department, Shri Vishnu Engineering college for women,AP)*
[2](*M. Tech, Associate Professor, ECE Department, Shri Vishnu Engineering college for women ,AP*)

ABSTRACT: *In this paper we propose a technique for software-implementation of an lifting with the goal of getting a customizable lifting DWT-core which can be used as a module in implementing a bigger system irrespective of ones choice of implementation platform. Here we have written the core processor Microblaze is designed in VHDL (VHSIC hardware description language), implemented using XILINX ISE 8.1 Design suite the algorithm is written in system C Language and tested in SPARTAN-3 FPGA kit by interfacing a test circuit with the PC using the RS232 cable. The test results are seen to be satisfactory. The area taken and the speed of the algorithm are also evaluated.*

Keywords: *lifting, Microblaze, Softcore, UART,VHDL*

I. Introduction

Lossy image compression can provide acceptable image quality while also providing dramatic reductions in image size. However, powerful methods for lossless image compression are still needed. Lossless compression can benefit the following important uses of imaging:

• Medical images,
• Seismic data,
• Satellite images,
• Manuscript images,
• Heavily edited images.

Accuracy in medical imagery is quite literally a matter of life and death. A doctor must discern small details in medical images to help correctly diagnose many diseases such as cancer. Thus, artifacts due to lossy compression cannot be tolerated. Seismic forces such as earthquakes are often preceded by small landscape changes prior to turning destructive, necessitating accurate imagery. Similarly, military decision makers analyzing satellite images require precise information in order to take informed actions based on this data. Manuscripts by famous authors can be preserved in their original state, thus preserving their study for future scholars. Heavily edited images benefit from lossless compression as many details can be gradually lost if an image is consistently saved in a lossy format. Generally, lossless image compression aids any application that must discern small details amongst a large data set with coding. This is because the DWT can decompose the signals into different sub-bands with both time and frequency information. It also supports features like progressive image transmission, compressed image manipulation, and region of interest coding. Recently several VLSI architectures have been proposed to realize single chip designs for DWT. Traditionally, such algorithms are implemented using programmable DSP chips for low-rate applications, or VLSI application specific integrated circuits (ASICs) for higher rates. In wavelet transforms, the original signal is divided into frequency resolution and time resolution contents. The decomposition of the image using 2-level DWT is shown in Figure1[1,2,3]

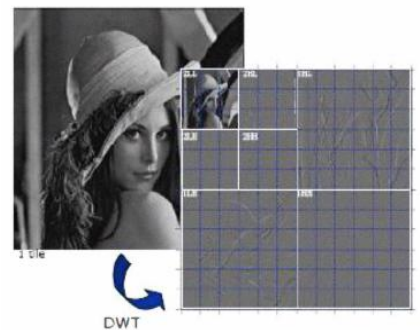


Fig 1. Decomposition of Image

II. Wavelet Transform

The best way to describe discrete wavelet transform is through a series of cascaded filters. We first consider the FIRbased discrete transform. The input image X is fed into a lowpass filter h' and a high-pass filter g' separately.The output of the twfilters are then sub sampled, resulting low-pass sub band yL and high-pass sub band yH .

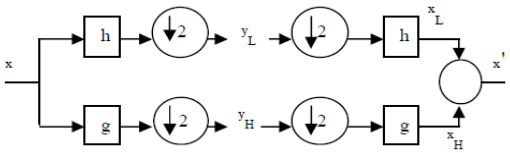


Fig 2. DWT analysis and synthesis system

The original signal can be reconstructed by synthesis filters h and g which take the up sampled yL and yH as inputs
[9]. To perform the forward DWT the standard uses a 1-D sub band decomposition of a 1-D set of samples into low-pass samples and high-pass samples. Low pass samples represent a down sampled low-resolution version of the original set. High-pass samples represent a down sampled residual versionof the original set, needed for the perfect reconstruction of the original set.

III. Design Flow

To build an embedded system on Xilinx FPGAs, the embedded development kit (EDK) is used to complete the reconfigurable design Figure 1 shows the design flow. Unlike the design flow in the traditional software design using C/C++ language or hardware design using hardware description languages, the EDK enables the integration of both hardware and software components of an embedded system.For the hardware side, the design entry from VHDL/Verilog is first synthesized into a gate-level netlist, and then translated into the primitives, mapped on the specific device resourcessuch as Look-up tables, flip-flops, and block memories.The location and interconnections of these device resources are then placed and routed to meet with the timing Constraints. A downloadable .bit file is created for the whole hardware platform. The software side follows thestandard embedded software flow to compile the source codes into an executable and linkable file (ELF) format. Meanwhile, a microprocessor software specification (MSS) file and amicroprocessor hardware specification (MHS) file are used to define software structure and hardware connection of the system. The EDK uses these files to control the design flow and eventually merge the system into a single downloadable file. The whole design runs on a real-time operating system
(RTOS).

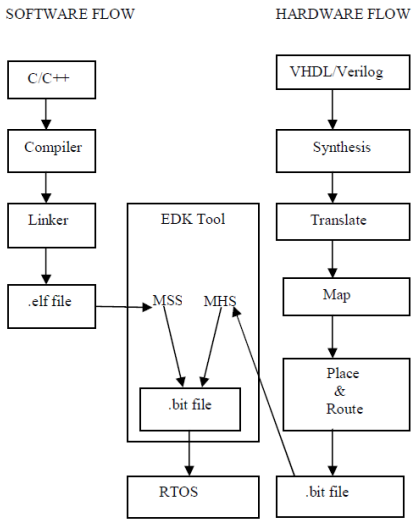


Fig 3. Design flow

The wavelet transform has proved to be an indispensable tool in data compression due to its ability to decorrelate data effectively and efficiently. The lifting scheme is a simple method for designing customized biorthogonal wavelets and offers several advantages:
• Allows a faster implementation of the wavelet transform,
• Saves storage by providing an in- place calculation of the wavelet transform,
• Simplifies determining the inverse wavelet transform,
• Provides a natural way to introduce and think about wavelets.

The main difference with classical constructions is that it does not rely on the Fourier transform . The main difference with classical constructions is that it does not rely on the Fourier transform. Due to this reformulation, second generation wavelets can be constructed. Second generation wavelets, unlike traditional, first generation wavelets, are not necessarily translates and dilates of a function. In this way, wavelets can be applied to non- smooth domains and curves or surfaces.

IV. VLSI Architecture of 2D-DWT

The proposed VLSI architecture shown in Fig.4.performs 2-D DWT with line- based method[2][6],which consists of five key modules:data choose module,th row DWT module,the column DWT module,DWT comtrol unit and external Ram.An N^2 / 4 external RAM is used to store the LL band output coefficients to carry out the multi-level decomposition,where N represents the width and the height of the input image.The DWT control unit controls the time sequence of the whole system.

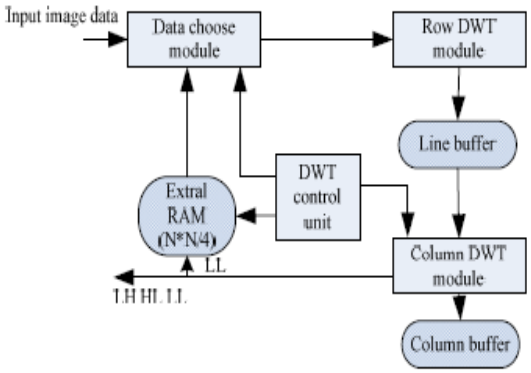


Fig 4. Proposed 2-D DWT architecture

First.one line if image data or LL sub-band data are routed in the Data Selector.Then the data enter into the row processor to perform !_D row DWT and the ouput data are stored in the line buffer.The number of the buffer is decided by the number of tap of the low-pass filter.When [(M+1)/2](M is the number of the tap of the low-pass filter)rows of data have finished the row DWT,the column DWT module starts to perform the column transform immediately and stores the intermediate results in the column buffer.The final transform data are stored in the external SRAM.

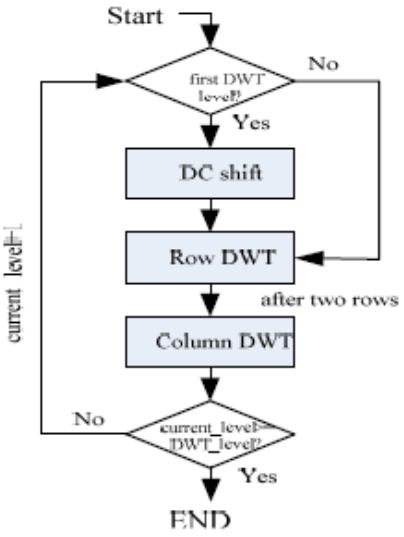


Fig5. FSM of DWT control unit

In this paper,we only discuss the wavelet transorm modules.The DWT control unit and the esternal RAM are not the purpose of this article.For the importance of the DWT unit,here we give out its FSM diagram shown in Fig.5.

V. General Concept of Lifting

The lifting transform at its highest level is very simple. The lifting transform can be performed via two operations: Predict and Update. Suppose we have the one dimensional signal a0. Lifting is done by performing the following sequence of operations:

1. Split a0 into Even-1 and Odd-1
2. d-1 = Odd-1 – Predict(Even-1)
3. a-1 = Even-1 + Update(d-1)

These steps are repeated to construct multiple scales of the transform. The inverse transformation is simple as well.

We only reverse the order of operations and change the signs. The even and odd sequences are then merged together to form the original signal. The wire diagram shows the forward transform visually. The coefficients a are represent the averages in the signal, while the coefficents in d represent the differences in the signal. Thus, these two sets also correspond to the low- pass and high- pass frequencies present in the signal. To mirror the operations of the wavelet transform, we must formulate the operations Predict and Update. For the Haar wavelet, we have the following steps:

• Predict(x) = x
• Update(y) = y / 2

The Haar case is very easy to derive and implement. Further, the operation can be performed on an input signal or image in- place , making it both time and space efficient. However, Haar has poor results due to the simplicity of the prediction step. Since every wavelet transform has a transform that can be formulated in terms of lifting steps, several other more effective transforms can be derived and used, such as Daubechies and symmetric biorthogonal.

Compression requires two major steps: decorrelation and coding. The process of lifting provides spatial decorrelation of image data, but no actual compression is performed by this step. Hence, a coding step must follow lifting to reduce the amount of data. An entropy coder, such as Huffman or arithmetic, can be used for this purpose.

VI. Lifting Dwt

The lifting scheme has been developed as a flexible tool suitable for constructing the second generation wavelet. It is composed of three basic operation stages: splitting, predicting, and updating. Fig.6 shows the lifting scheme of the wavelet filter computing one dimension signal:

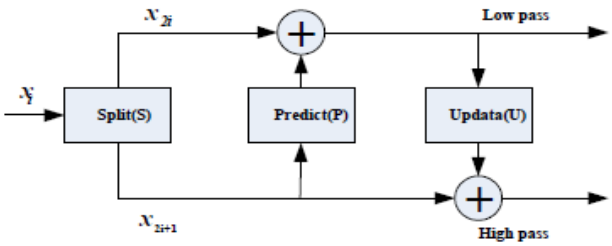


Fig 6. Block diagram of the lifting scheme.

Splitting Step

The number of samples can be reduced by simply subsampling the input signal:

a-1,k = a0,2k

where a0,2k is an even sample (2k) at the starting level (0) of the transformation and a-1,k is the sample at the next level (-1). Similarly, the other samples are partitioned as

d-1,k = a0,2k+1

where d - 1,k are the difference, or wavelet, coefficients. This partition corresponds to the so- called Lazy wavelet. This will clearly not decorrelate the signal (unless of course the coefficients are already small, which is an unreasonable assumption). While this step does not achieve much, the next steps, predict and update, will show more explicitly how this partitioning method can be used to achieve our goal of decorrelating a signal in a recoverable manner.

Prediction Step

This section covers the prediction stage. Three main topics are discussed here. First, we discuss the main idea behind prediction and how this can lead us to linear prediction. Next, we generalize this type to other functions by deriving the steps for cubic prediction. Finally, we discuss the problems that can occur along the boundaries of signals and how to determine filter coefficients, which are used for prediction.

Update Step

We have one step left: update. The goal at this stage is to maintain some global properties of the original signal in the reduced set. For example, if the signal is a 2D image, we would like the average pixel intensity value to be the same across all scales of the transform, which implies that the final coefficient at the last scale is the average pixel value of the entire, original image. We design a system to preserve some fixed Ñ 10 moments of the wavelet function at each level. Ñ is referred to as the number of real vanishing moments. The first Ñ moments are given by At each scale, we want to preserve up to Ñ- 1 of the a's at every level and use this information to see how much of every d coefficient is needed to update every a. The coefficients used for this update procedure are named lifting coefficients.

VII. Microblaze Processor design

FIELD-PROGRAMMABLE GATE ARRAYS (FPGA's) are flexible and reusable high-density circuits that can be easily re-configured by the designer, enabling the VLSIdesign / validation /simulation cycle to be performed more quickly and less expensive. Increasing device densities have prompted FPGA manufacturers, such as Xilinx and Altera, to incorporate larger embedded components, including multipliers, DSP blocks and even embedded processors. One of the recent architectural enhancements in the Xilinx Spartan, Virtex family architectures is the introduction of the MicroBlaze (Soft IP) and PowerPC405 hard-core embeddedprocessor (1l].The MicroBlaze processor is a 32-bit Harvard Reduced Instruction Set Computer (RISC) architecture optimized for implementation in Xilinx FPGAs with separate 32-bit instruction and data buses running at full speed to execute programs and access data from both on-chip and external memory at the same time.

7.1Background

The backbone of the architecture is a single-issue, 3-stage pipeline with 32 general-purpose registers (does not have any address registers like the Motorola 68000 Processor), an Arithmetic Logic Unit (ALU), a shift unit, and two levels of interrupt. This basic design can then be configured with more advanced features to tailor to the exact needs of the target embedded application such as: barrel shifter, divider, multiplier, single preCIsIon floating-point unit (FPU), instruction and data caches, exception handling, debug logic, Fast Simplex Link (FSL) interfaces and others.

This flexibility allows the user to balance the required performance of the target application against the logic area cost of the soft processor MicroBlaze also supports reset, interrupt, user exception, and break hardware exceptions. For interrupts, MicroBlaze supports only one external interrupt source (connecting to the Interrupt input port) (2). If multiple interrupts are needed, an interrupt controller must be used to handle multiple interrupt requests to MicroBlaze shown in figure7.

An interrupt controller is available for use with the Xilinx Embedded Development Kit (EDK) software tools. The processor will only react to interrupts if the Interrupt Enable (IE) bit in the Machine Status Register (MSR) is set to 1. On an interrupt the instruction in the execution stage will complete, while the instruction in the decode stage is replaced by a branch to the interrupt vector (address Ox 10). The interrupt return address (the PC associated with the instruction in the decode stage at the time of the interrupt) is automatically loaded into general-purpose register. In addition, the processor also disables future interrupts by clearing the IE bit in the MSR. The IE bit is automatically set again when executing the RTlD instruction.

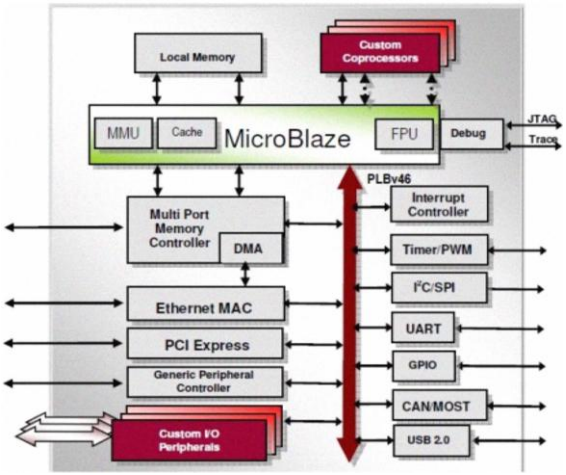


Fig 7. Microblaze Architecture Block Diagram

Due to the advancement in the fabrication technology and the increase in the density of logic blocks on FPGA, the use of FPGA is not limited anymore to debugging and prototyping digital electronic circuits. Due to

the enormous parallelism achievable on FPGA and the increasing density of logic blocks, it is being used now as a replacement to ASIC solutions in a few applications where the time to market is critical and also entire embedded processor systems are implemented on these devices with soft core processors embedded in the system. Soft cores are technology independent and require only simulation and timing verification after synthesized to a target technology. This reduces the design cycle development time by a major factor as compared to the development cycle for a hard core processor and has the advantage of customizing the soft core design for a specific application.

7.2.EXPERIMENTAL SETUP

7.2.1. Xilinx Platform Studio

The Xilinx Platform Studio (XPS) is the development environment or GUI used for designing the hardware portion of your embedded processor system. B. Embedded Development Kit Xilinx Embedded Development Kit (EDK) is an integrated software tool suite for developing embedded systems with Xilinx MicroBlaze and PowerPC CPUs. EDK includes a variety of tools and applications to assist the designer to develop an embedded system right from the hardware creation to final implementation of the system on an FPGA. System design consists of the creation of the hardware and software components of the embedded processor system and the creation of a verification component is optional. A typical embedded system design project involves: hardware platform creation, hardware platform verification (simulation), software platform creation, software application creation, and software verification. Base System Builder is the wizard that is used to automatically generate a hardware platform according to the user specifications that is defmed by the MHS (Microprocessor Hardware Specification) file. The MHS file defines the system architecture, peripherals and embedded processors]. The Platform Generation tool creates the hardware platform using the MHS file as input.

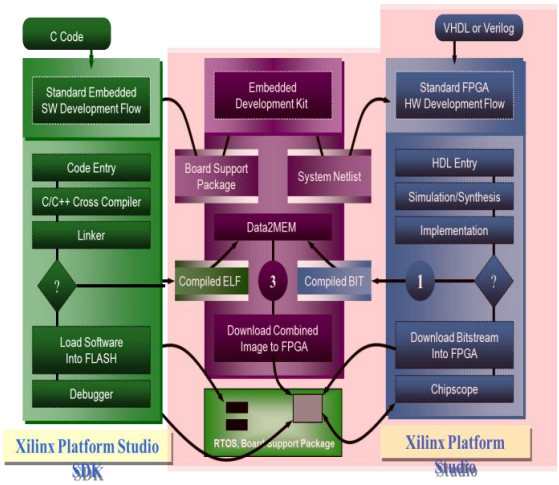


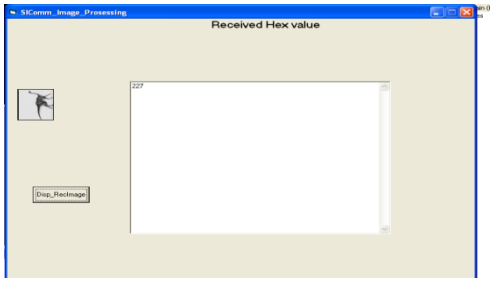
Fig 8. Embedded Development Kit Design Flow

The creation of the verification platform is optional and is based on the hardware platform. The MHS file is taken as an input by the Simgen tool to create simulation files for a specific simulator. Three types of simulation models can be generated by the Simgen tool: behavioral, structural and timing models. Some other useful tools available in EDK are Platform Studio which provides the GUI for creating the MHS and MSS files. Create / Import IP Wizard which allows the creation of the designer's own peripheral and import them into EDK projects. Bitstream Initializer tool initializes the instruction memory of processors on the FPGA shown in figure2. GNU Compiler tools are used for compiling and linking application executables for each processor in the system [8]. There are two options available for debugging the application created using EDK namely: Xilinx Microprocessor Debug (XMD) for debugging the application software using a Microprocessor Debug Module (MDM) in the embedded processor system, and Software Debugger that invokes the software debugger corresponding to the compiler being used for the processor. C. Software Development Kit Xilinx Platform Studio Software Development Kit (SDK) is an integrated development environment, complimentary to XPS, that is used for C/C++ embedded software application creation and verification.The software application can be written in a "C or C++" then the complete embedded processor system for user application will be completed, else debug & download the bit file into FPGA. Then FPGA behaves like processor implemented on it in a Xilinx Field Programmable Gate Array (FPGA) device.

VIII. Experimental Results

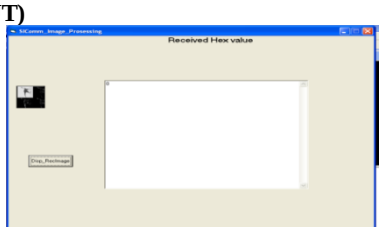
Experiments are performed on gray level images to verify the proposed method. These images are represented by 8 bits/pixel and size is 128 x 128. Image used for experiments are shown in below figure.

INPUT IMAGE

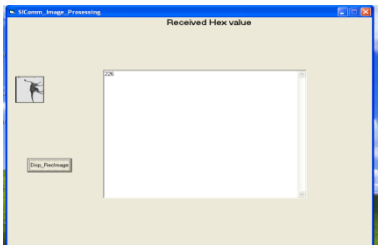


The measurands used for proposed method are as follows:
An often used global objective quality measure is the mean square error (MSE) defined as
MSE=((i/p-o/p)^2)/n*m
Where, nxm is the number of total pixels. f (i,j) and f(i,j)' are the pixel values in the original and reconstructed image. The
peak to peak signal to noise ratio (PSNR in dB) [9-11] is
calculated as

PSNR=Progressive SNR
=10log(255^2/MSE))
Usable gray level values range from 0 to 255.
And the Compressed Image is showed in below Figure.
COMPRESSED IMAGE(DWT OUTPUT)



REVERSE DWT



Tabulation Results
The Algorithm is implemented in Microblaze Processor and the results are furnished in the tabulation below
And the synthesis report is below

Post Synthesis Clock Limits

These are the post synthesis clock frequencies. The critical frequencies are marked with green.
The values reported here are post synthesis estimates calculated for each individual module. These values will change after place and route is performed on the entire system.

MODULE	CLK Port	MAX FREQ
microblaze_0	FSL3_S_CLK	83.639MHz
microblaze_0	DBG_CLK	83.639MHz
SRAM_256Kx32	OPB_Clk	111.247MHz
RS232	OPB_Clk	139.159MHz
debug_module	debug_module/drck_i	143.390MHz
debug_module	OPB_Clk	143.390MHz
debug_module	bscan_update	143.390MHz
mb_opb	OPB_Clk	178.891MHz
ilmb	LMB_Clk	294.291MHz
dlmb	LMB_Clk	294.291MHz

Device Utilization Summary:

Number of BSCANs	1 out of 1	100%
Number of BUFGMUXs	2 out of 8	25%
Number of DCMs	1 out of 4	25%
Number of External IOBs	62 out of 97	63%
Number of LOCed IOBs	62 out of 62	100%
Number of MULT18X18s	3 out of 12	25%
Number of RAMB16s	4 out of 12	33%
Number of Slices	1174 out of 1920	61%
Number of SLICEMs	180 out of 960	29%

IX. CONCLUSION

In this paper, a DWT-based reconfigurable system is designed using the EDK tool. Hardware architectures of two dimensional (2-D) DWT have been implemented as a coprocessor in an embedded system. One is direct implementation of 2-D DWT by cascading two 1-D DWTs.Another is 2-D DWT implementation with control and architecture optimization. In addition, the hardware cost of these two architectures is compared for benchmark images. This type of work using EDK can be extended to other applications of embedded system. These two architectures applications compared for benchmark images. This type of work using EDK can be extended to other applications of embedded systems.

References

[1] David S. Taubman, Michael W. Marcellin - JPEG 2000 – Image compression, fundamentals, standards andpractice", Kluwer academic publishers, Second printing - 2002.

[2] G. Knowles, "VLSI Architecture for the Discrete Wavelet Transform," Electronics Letters, vo1.26, pp. 1184-1185,1990.

[3] M, Vishwanath, R. M. Owens, and M. 1. Irwin, "VLSI Architecturesfor the Discrete Wavelet Transform," IEEE Trans. Circuits And Systems II, vol. 42, no. 5, pp. 305-316, May. 1995.

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

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

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

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

[8] Xilinx Inc. Xilinx ISE and Xilinx EDK tools.

[9] Xilinx. Inc., Platform Specification Format Reference Manual, Embedded Development Kit EDK 9.2i

[10] Xilinx, Embedded System Example, XAPP433, version 2.2, 2006.

[11] Xilinx Inc. MicroBlaze Reference Manual, version 10.1.

[12] Xilinx Inc. Xilinx ISE and Xilinx EDK tools.

[13] Spartan-3 Starter Kit Board User Guide, Xilinx, Inc.

[14] Embedded System Tools Reference Manual, Xilinx, Inc

[15] Spartan-3 FPGA Family: Complete Data Sheet

[16]. Dr.K.Veera swamy, Dr.B.Chandra Mohan,Y.V.Bhaskar Reddy and Dr.S.Srinivasa Kumar, Image Compression and Watermarking scheme using Scalar Quantization. IJNGN, 2000.

[17]. B.Shrestha, Dr.Charles,G.O'Hara and Dr. Nicolas H. Younan, JPEG2000:Image Quality metrics.ASPRS,2005.