Chapter 9

Implementation of Lapped Biorthogonal Transform for JPEG-XR Image Coding

Muhammad Riaz ur Rehman, Gulistan Raja and Ahmad Khalil Khan

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/53100

1. Introduction

Advancements in digital image devices have culminated in increase in image size and quality. High quality digital images are required in different fields of life for example in medical, surveillance, commercials, space imaging, mobile phones, play stations and digital cameras. As a result, memory requirement for storing these high quality images has been increased enormously. Moreover, if we want to transmit these images over communication channel, it will require high bandwidth. Thus there is a need to develop techniques that reduces the size of image without significantly compromising the quality of digital image so that it can be stored and transmitted efficiently.

Compression techniques exploit redundancy in image data to reduce the required amount of storage for image. Different compression performance parameters such as compression ratio, computation complexity, compression / decompression time and quality of compressed image vary with different compression techniques. Most widely used image compression standard is JPEG (ISO/IEC IS 10918-1 | ITU-T T.81) [1]. It supports baseline, hierarchical, progressive and lossless modes and provides high compression at low computational cost. Figure 1 shows steps in JPEG encoding. It uses Discrete Cosine Transform (DCT) which is applied on 8x8 image block. However at low bit rate it produces blocking artifacts.

To overcome the limitations of JPEG, new standard i.e. JPEG2000 (ISO/IEC 15444-1 | ITU-T T.800) was developed [2]. JPEG2000 uses Discrete Wavelet Transform (DWT) and provides high compression ratio without compromising the quality of image quality even at low bit rates. It supports lossless, lossy, progressive and region of interest encoding. However, these advantages are achieved at the cost high computational complexity. Therefore there was a

need for a compression technique that not only preserves the quality of high resolution images but also keep the storage and computational cost as low as possible.

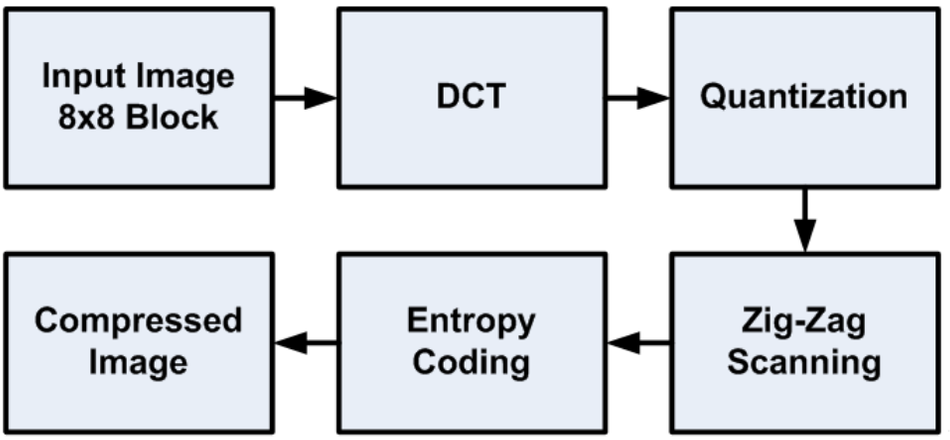

Figure 1. JPEG Encoding

A new image compression standard, JPEG eXtended Range (JPEG XR) has been developed which addresses the limitations of currently used image compression standards [3-4]. JPEG XR (ITU-T T.832 | ISO/IEC 29199-2) mainly targets to increase the capabilities of exiting coding techniques and provides high performance at low computational cost. JPEG XR compression stages are almost same at higher level as compared to existing compression standards but lower level operations are different such as transform, quantization, scanning and entropy coding techniques. It supports lossless as well as lossy compression. JPEG XR compression stages are shown in Figure 2.

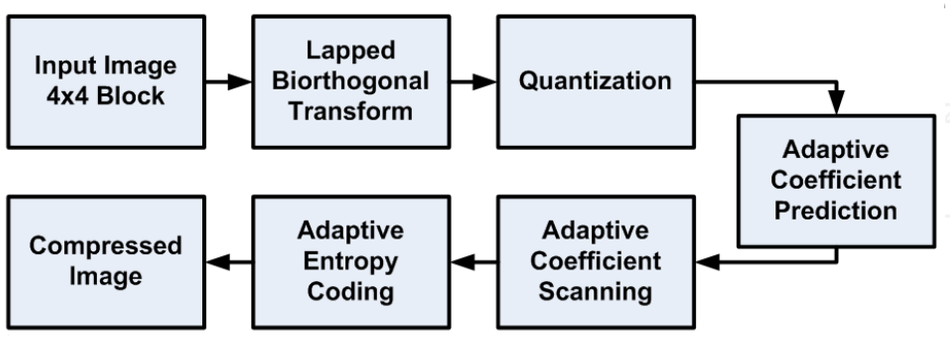

Figure 2. JPEG XR Encoding

JPEG XR use Lapped Biorthogonal Transform (LBT) to convert image samples into frequency domain coefficients [5-8]. LBT is integer transform and it is less computationally expensive than DWT used in JPEG2000. It reduces blocking artifacts at low bit rates as compared to JPEG. Thus due to less computational complexity and reduced artifacts, it significantly

http://dx.doi.org/10.5772/53100

improves the overall compression performance of JPEG XR. Implementation of LBT can be categorized into software based implementation and hardware based implementation. Software based implementation is generally used for offline processing and designed to run on general purpose processors. Performance of software based implementation is normally less than hardware based implementation and mostly it is not suitable for real time applications. Hardware based implementation provide us superior performance and mostly suitable for real time embedded applications. In this chapter we will discuss LabVIEW based software implementation and Micro Blaze based hardware implementation of LBT. Next section describes the working of Lapped Biorthogonal Transform.

2. Lapped Biorthogonal Transform (LBT)

Lapped Biorthogonal Transform (LBT) is used to convert image samples from spatial domain to frequency domain in JPEG XR. Its purpose is the same as discrete cosine transform (DCT) in JPEG. LBT in JPEG XR is operated on 4x4 size image block. LBT is applied on blocks and macro blocks boundaries. Input image is divided into tiles prior to applying LBT in JPEG XR. Each tile is further divided into macro blocks as shown in Figure 3.

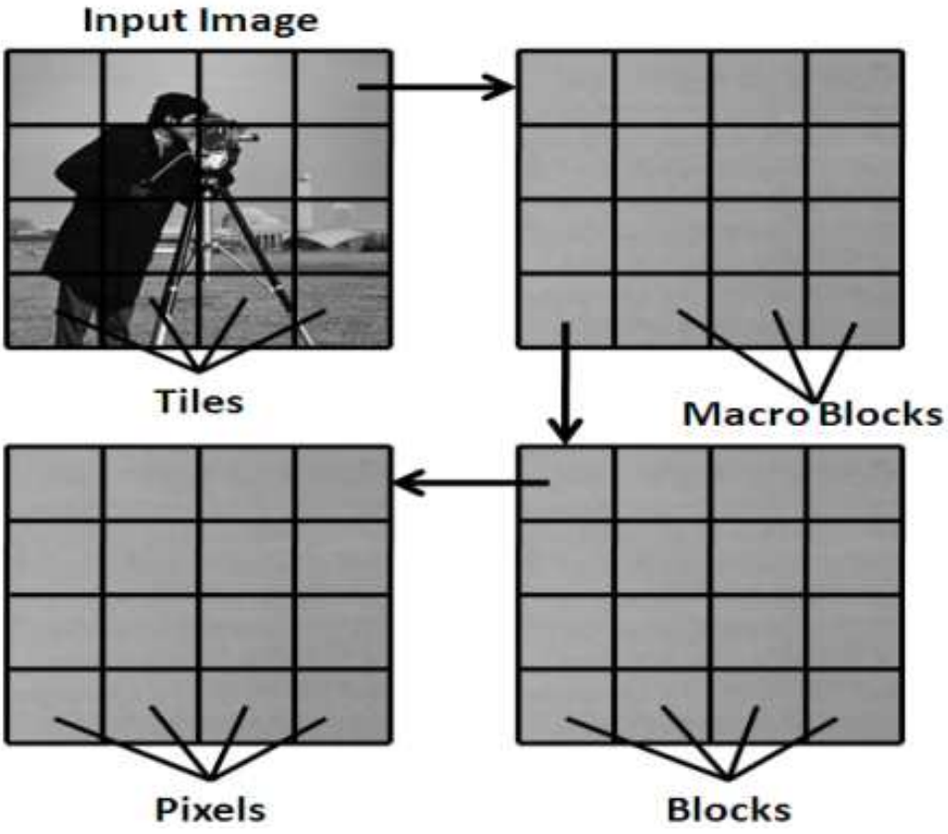

Figure 3. Image Partitioning

Each macro block is a collection of 16 blocks while a block is composed of 16 image pixels. Image size should be multiple of 16; if size is not multiple of 16, then we extend the height and or width of image to make it multiple of 16. This can be done by replicating the image sample values at boundaries. Lapped Biorthogonal Transform consists of two key operations:

1. Overlap Pre Filtering (OPF)
2. Forward Core Transform (FCT)

Encoder uses OPF and FCT operations in following steps as shown in Figure 4.

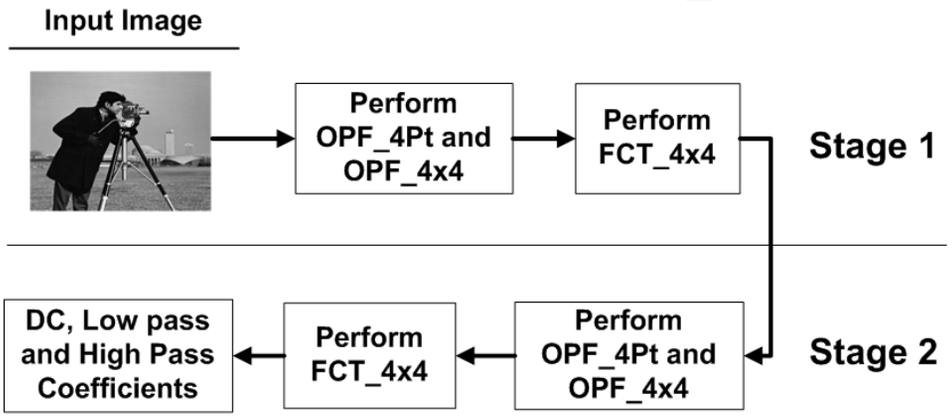

Figure 4. Lapped Biorthogonal Transform Stages [9]

OPF is applied on block boundaries, areas of sizes 4x4, 4x2 and 2x4 between block boundaries are shown in Figure 5.

The various steps performed in LBT are as follows:

1. In Stage 1, Overlap pre filter (OPF_4pt) is applied to 2x4 and 4x2 areas between blocks boundaries. Additional filter (OPF_4x4) is also applied to 4x4 area between block boundaries.
2. A forward core transform (FCT_4x4) is applied to 4x4 blocks. This will complete stage 1 of LBT.
3. Each 4x4 block has one DC coefficient. As macro block contains 16 blocks so we have 16 DC coefficients in one macro block. Arrange all 16 DC coefficients of macro blocks in 4x4 DC blocks.
4. In stage 2, Overlap pre filter (OPF_4pt) is applied to 2x4 and 4x2 areas between DC blocks boundaries. Additional filter (OPF_4x4) is also applied to 4x4 area between DC block boundaries.
5. Forward core transform (FCT_4x4) is applied to 4x4 DC blocks to complete stage 2 of LBT. This will results in one DC coefficient, 15 low pass coefficients and 240 high pass coefficients per macro block.

http://dx.doi.org/10.5772/53100

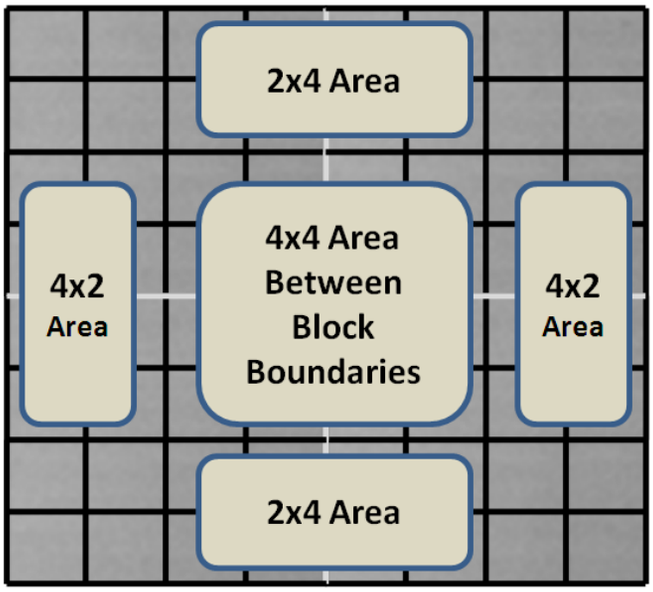

Figure 5. Image partitioning

The 2-D transform is applied to process the two dimensional input image. A 2-D transform is implemented by performing 1-D transform in rows and columns of 2-D input image. Matrix generated by Kronecker product is also used to obtain 2-D transform. Transform Y of 2-D input image X is given by Eq. (1) and Eq. (2):

$$Y = MX \tag{1}$$

$$M = Kron(T1,\ T2) \tag{2}$$

Where T1 and T2 are 1-D transform matrix for rows and columns respectively. Forward Core Transform is composed of Hadamard transform, Todd rotation transform and Todd-dodd rotation transform. Hadamard transform is Kronecker product of two 2-point hadamard transform Kron(Th, Th) where Th is given by Eq. (3):

$$Th = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3}$$

Todd rotation transform is Kronecker product of 2-point Hadamard transform and 2-point rotation transform Kron (Th, Tr) where Tr is given by Eq. (4):

$$Tr = \frac{1}{\sqrt{4+2\sqrt{2}}}\begin{bmatrix} 1+\sqrt{2} & 1 \\ 1 & -\left(1+\sqrt{2}\right) \end{bmatrix} \tag{4}$$

Toddodd rotation transform is Kronecker product of two 2-point rotation transform Kron (Tr, Tr). Overlap pre filtering is composed of hadamard transform Kron (Th, Th), inverse hadamard transfom, 2-point scaling transform Ts, 2-point rotation transform Tr and Toddodd transform Kron (Tr, Tr). Inverse hadamard transform is Kronecker product of two 2-point inverse hadamard transform Kron (inverse (Th), inverse (Th)).

3. LabVIEW based Implementation of LBT

LabVIEW is an advanced graphical programming environment. It is used by millions of scientists and engineers to develop sophisticated measurement, test, and control systems. It offers integration with thousands of hardware devices. It is normally used to program, PXI based system for measurement and automation. PXI is a rugged PC-based platform for measurement and automation systems. It is both a high-performance and low-cost deployment platform for applications such as manufacturing test, military and aerospace, machine monitoring, automotive, and industrial test. In LabVIEW, programming environment is graphical and it is known as virtual instrument (VI).

LabVIEW implementation of LBT consists of 10 sub virtual instruments (sub-VIs). LBT implementation VI hierarchy is shown in Figure 6.

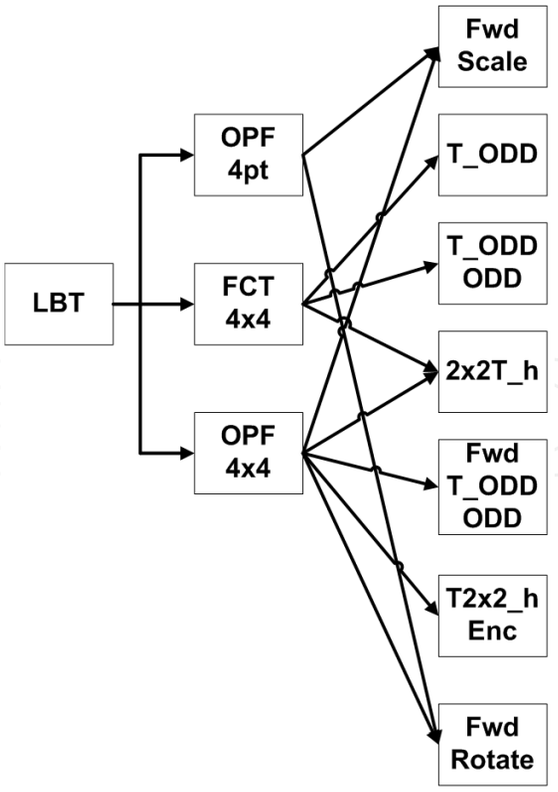

Figure 6. LBT VI Hierarchy

http://dx.doi.org/10.5772/53100

These sub-VIs are building blocks of LBT. Operations of these sub-VIs are according to JPEG XR standard specifications [3]. OPF 4pt, FCT 4x4, OPF 4x4 are main sub-VIs and are used in both stages of LBT. OPF 4pt further uses FWD Rotate and FWD Scale VIs. Similarly FCT 4x4 and OPF 4x4 require T_ODD, 2x2T_h, T_ODD ODD, T2x2h_Enc, FWD_T ODD ODD sub-VIs.

Figure 7 shows main block diagram of LBT implementation in LabVIEW that performs sequence of operations on the input image.

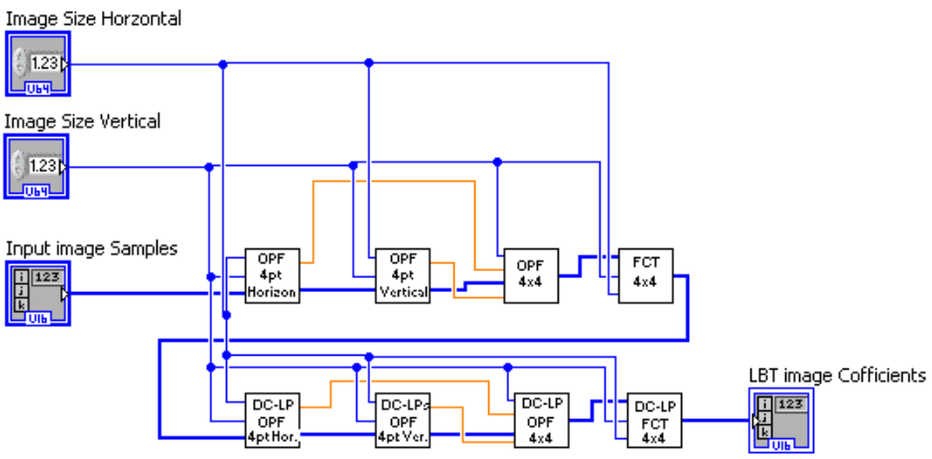

Figure 7. LBT Block Diagram

In stage 1, image samples are processed by OPF 4pt in horizontal direction (along width) of the image. This operation is performed on 2x4 boundary areas in horizontal direction. Figure 8 shows block diagram of OPF 4pt.

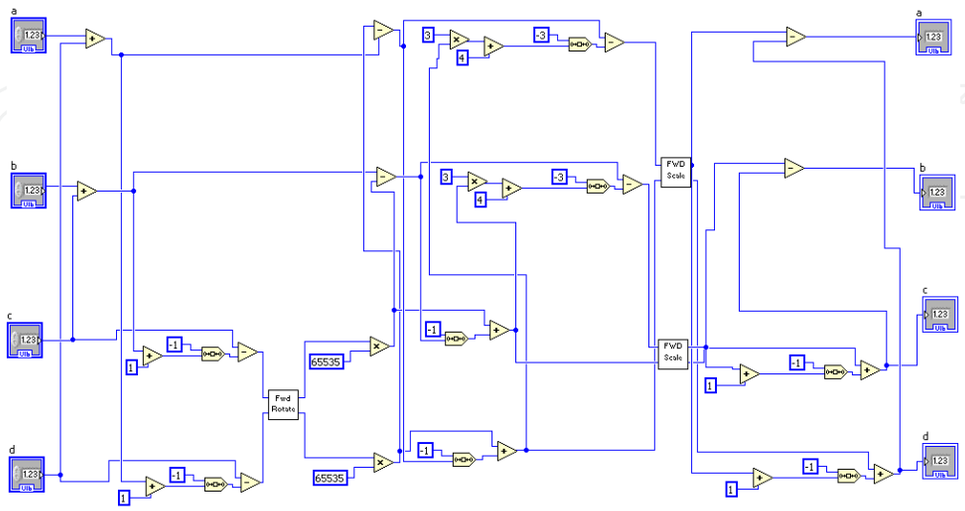

Figure 8. OPF 4pt Block Diagram

Each OPF 4pt performs addition, subtraction, multiplication and logical shifting on four image samples. The OPF 4pt requires four image samples and process them in parallel. For example, addition of samples a, d and b, c are performed in parallel as shown in Figure 8. Data is processed simultaneously when it is available to operators: addition, subtraction, multiplication or logical shifter. This parallel computation speeds up the overall execution time. It uses two additional sub-VIs i.e., Fwd Rotate and Fwd Scale. These sub-VIs require two image samples and can be executed in parallel. In OPF 4pt, two Fwd Scale sub VIs are executed in parallel. Two OPF 4pt sub-VIs are required for 2x4 and 4x2 block boundaries areas. Figure 9 shows processing of OPF 4pt.

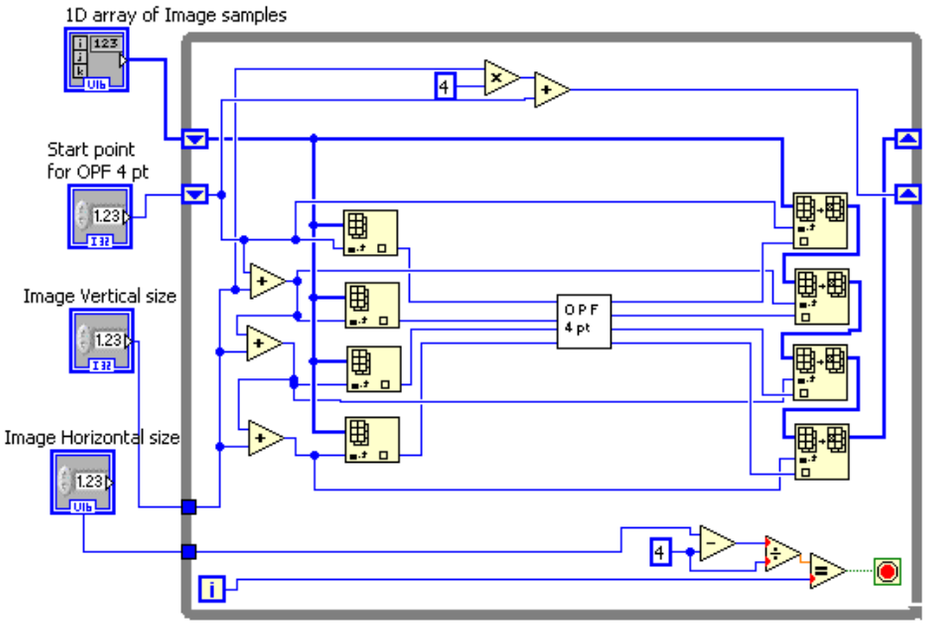

Figure 9. Block Diagram for OPF 4pt Processing

OPF 4pt operation is also performed in vertical direction (along height) of the image. For processing in both directions OPF 4pt requires 1D array of input image samples, starting point for the operation of OPF 4pt and dimensions of input image.

After the operation of OPF 4pt, OPF 4x4 is performed on 4x4 areas between block boundaries to complete overlap pre filtering. Figure 10 shows block diagram of OPF 4x4.

OPF 4x4 operates on 16 image samples. It uses T2x2_Enc, FWD Rotate, FWD Scale, FWD ODD and 2x2T_h sub-VIs. Here these sub-VIs are also executes in parallel. Four T2x2h_Enc and 2x2T_h sub-VIs are executing in parallel. Similarly FWD Rotate, FWD Scale and FWD ODD are also executed in parallel. OPF 4x4 starts processing on 16 image samples at once and outputs all 16 processed image samples at same time. Figure 11 shows block diagram for processing of OPF 4x4.

For processing of image samples for OPF 4x4 operation: start point of OPF 4x4 and image dimensions are required along with input images samples. After the processing of OPF 4x4, FCT 4x4 is performed on each 4x4 image block. Figure 12 shows block diagram of FCT 4x4.

http://dx.doi.org/10.5772/53100

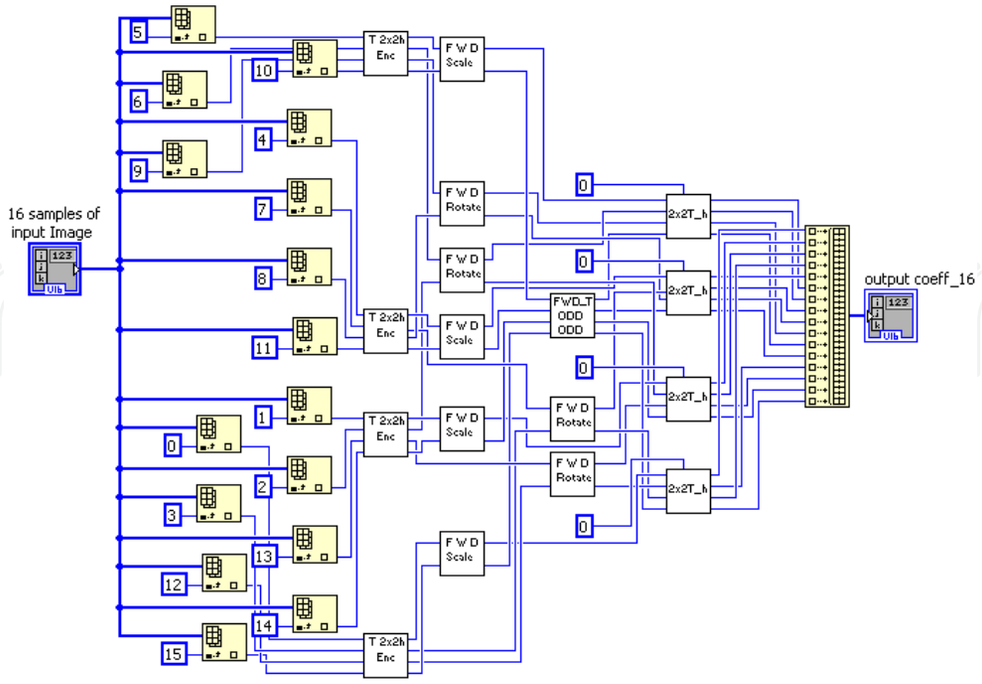

Figure 10. Block Diagram of OPF 4x4

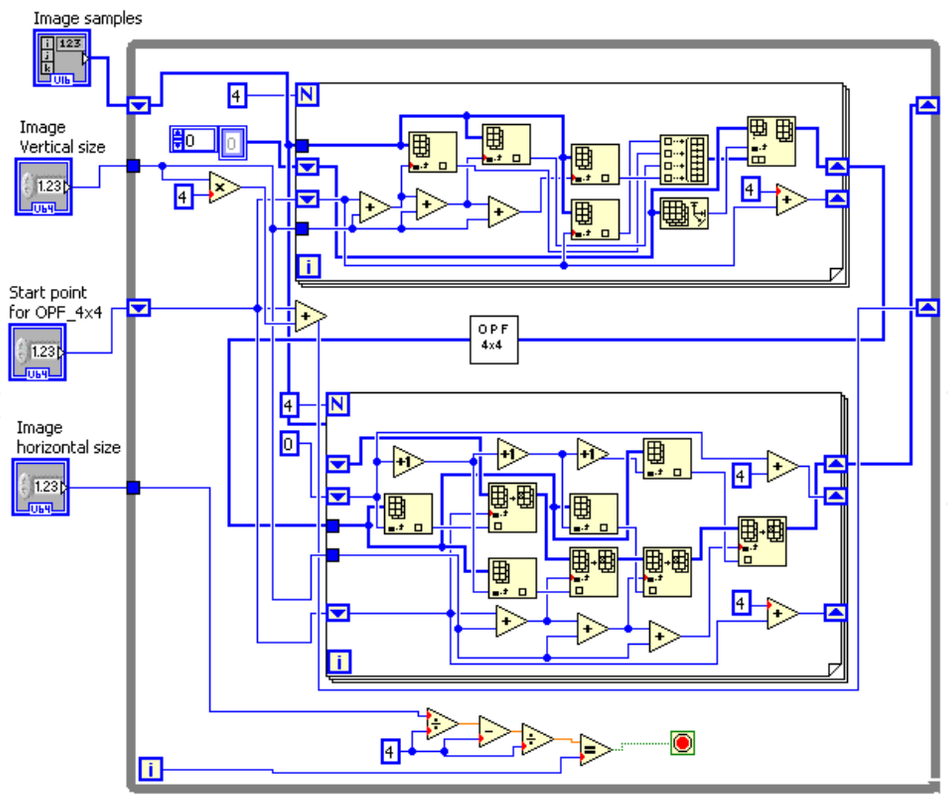

Figure 11. Block Diagram for OPF 4x4 Processing

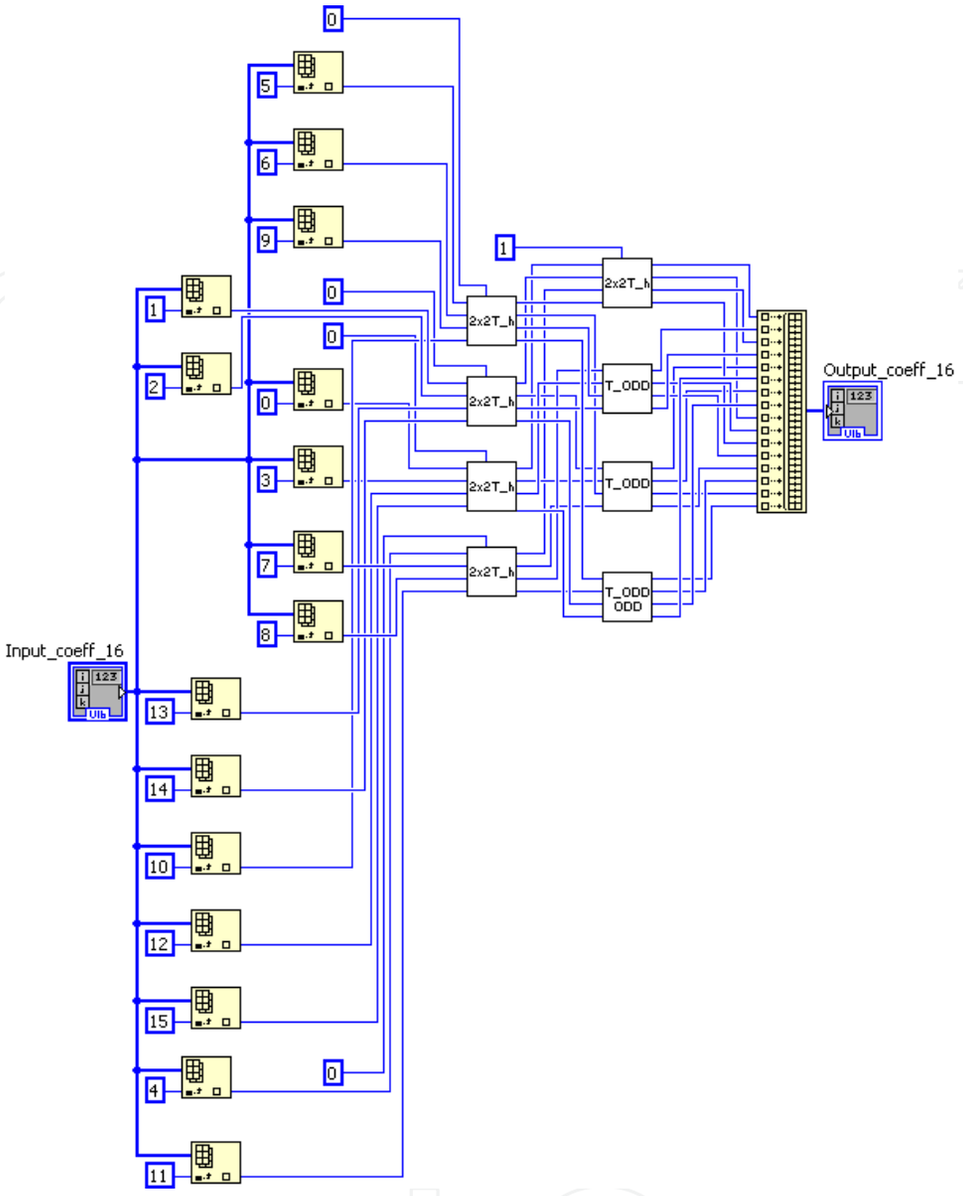

Figure 12. Block Diagram of FCT 4x4

FCT 4x4 operation requires 2x2T_h, T_ODD and T_ODDODD sub-VIs. These sub-VIs are also executed in parallel to speed up the operation of FCT 4x4. It is operated on 16 image samples that are processed in parallel. This completes the stage 1 of LBT. This will result one DC coefficient in each 4x4 block. In stage 2, all operations will be performed on these DC coefficients of all blocks. DC coefficients will be considered as image samples and arranged in 4x4 blocks. OPF 4pt is performed in horizontal and vertical directions on DC coefficients block boundaries with 4x2 and 2x4 areas. OPF 4x4 is also applied on 4x4 areas between DC blocks boundaries. FCT 4x4 is performed on each DC 4x4 blocks to complete stage 2 of LBT. At this stage, each macro block contains 1 DC, 15 low pass coefficients and 240 high pass coefficients.

http://dx.doi.org/10.5772/53100

We tested LabVIEW implementation on NI-PXIe 8106 embedded controller. It has Intel 2.16GHz Dual core processor with 1GB RAM. It takes 187.36 ms to process test image of size 512x512. We tested LBT in lossless mode. Functionality of implementation is tested and verified with JPEG XR reference software ITU-T T835 and standard specifications ITU-T T832.Memory usage by top level VI is shown in Table 1.

Resource Type	Used
Front panel Objects	22.6 KB
Block Diagram Objects	589.4 KB
Code	73.7 KB
Data	66.6 KB
Total	752.2 KB

Table 1. Memory Usage

Important parameters of implementation of top level VI and sub-VIs are shown in Table 2.

VI	No. of Nodes	Wire Sources	Connector Inputs	Connector Outputs
LBT.vi	561	641	0	0
OPF 4pt.vi	61	60	4	4
OPF 4x4.vi	56	90	1	1
FCT 4x4.vi	48	71	1	1
Fwd Scale.vi	28	26	2	2
Fwd Rotate.vi	14	12	2	2
2x2 T_h.vi	19	15	5	4
FWD T_ODD ODD.vi	41	37	4	4
T2x2h Enc.vi	25	21	4	4
T_ODD ODD.vi	45	41	4	4
T_ODD.vi	58	54	4	4

Table 2. VIs Parameters

4. Soft processor based hardware design of LBT

To use Lapped Biorthogonal transform in real time embedded environment, we need its hardware implementation. Application specific hardware for LBT provides excellent performance but up-gradation of hardware design is difficult because it requires remodeling of whole hardware design. Pipeline implementation of LBT also provides outstanding performance but due to sequential nature of LBT, it requires large amount of memory usage [10-12]. In this section, we describe a soft embedded processor based implementation of LBT. The proposed architecture design is shown in Figure 13.

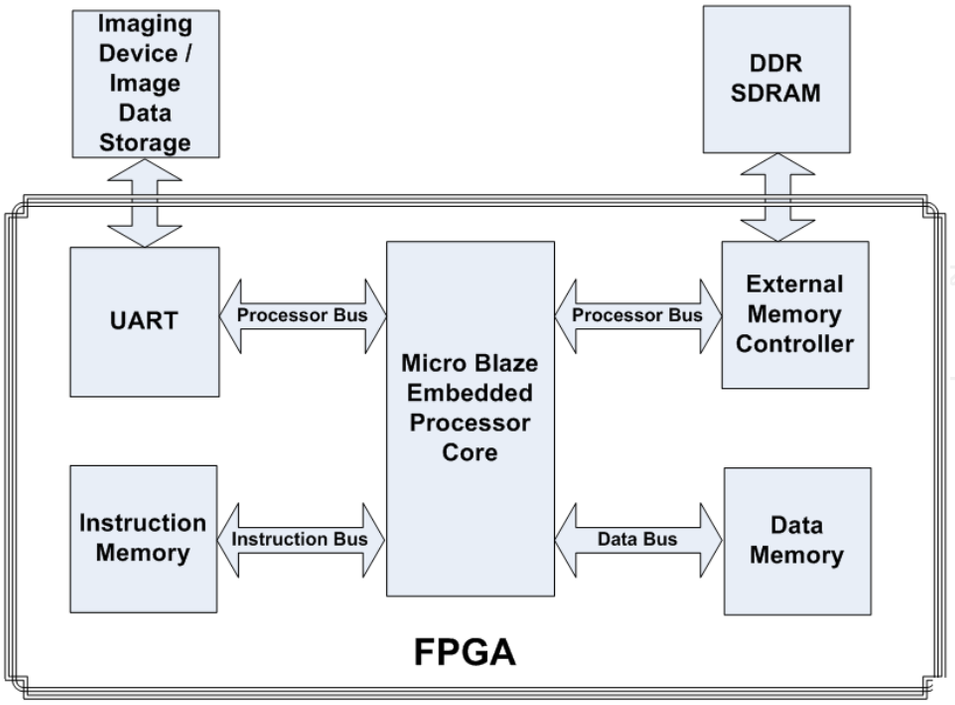

Figure 13. Proposed Architecture Design

Soft embedded processor is implemented on FPGA and its main advantage is that we can easily reconfigure or upgrade our design. The processor is connected to UART and external memory controller through processor bus. Instruction and data memories are connected to soft embedded processor through instruction and data bus respectively. The instructions of LBT processing are stored in instruction memory that will be executed by the proposed soft embedded processor core. Block RAM (BRAM) of FPGA is used as data and instruction memory.

For the processing of LBT, digital image is loaded into DDR SDRAM from external source like imaging device through UART. Image is first divided into fix size tiles i.e. 512x512. Tile data is fetched from DDR SDRAM into the data memory. Each tile is processed independently. OPF_4pt and OPF_4x4 operations are applied across blocks boundaries. After that FCT_4x4 operation is applied on each block to complete first stage of LBT. At this stage, each block has one DC coefficient.

For second stage of LBT, we consider these DC coefficients as single pixel arranged in DC blocks of size 4x4 and same operations of stage 1 are performed. After performing OPF_4pt, OPF_4x4 and FCT_4x4, stage 2 of LBT is completed. At this stage, each macro block has 1 DC coefficient, 15 low pass coefficients and 240 high pass coefficients. We send these coefficients back to DDR SDRAM and load new tile data into data memory. DDR SDRAM is just used for image storage and can be removed if streaming of image samples from sensor is available. Only data and instruction memory is used in processing of LBT. Flow diagram in Figure 14 gives summary of operations for LBT processing.

http://dx.doi.org/10.5772/53100

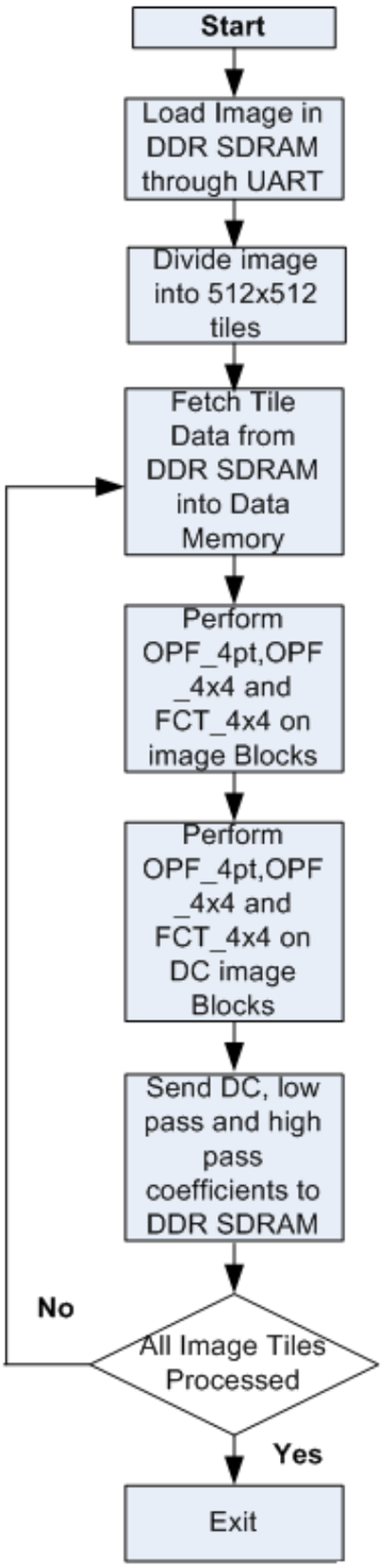

Figure 14. Flow Diagram of LBT Processing in Proposed Design [9]

The proposed design is tested on Xilinx Virtex-II Pro FPGA and verified the functionality of design according to standard specifications ITU-T T832 and reference software ITU-T T835. Test Image is loaded into DDR SDRAM through UART from computer. Same test image is also processed by reference software and compares the results. Both processed images were same when indicates correct functionality of our design. FPGA resources used in implementation are shown in Table3.

Resource Type	Used	% age of FPGA
Number Slice Registers	3,742	13%
Number of occupied Slices	3,747	27%
Number of 4 input LUTs	2,962	10%
Number of RAMB16s	25	18%
Number of MULT18X18s	3	2%

Table 3. FPGA Resource Utilization

Processor specifications of design are listed in Table 4.

Processor Speed	100MHz
Processor Bus Speed	100MHz
Memory for Instruction and Data	32KB

Table 4. Processor Resources

Memory required for data and instruction in our design is 262,144 bits. As the input image is divided into fix size tiles i.e. 512x512, design can process large image sizes. Minimum input image size is 512 x 512. Due to less memory requirements, easy up-gradation and tile based image processing. It is suitable for low cost portable devices. Test image is used of size 512x512 and in unsigned-16 bit format. Execution time to process test image is 27.6ms. Compression capability for test image is 36 frames per second. Figure 15 shows original and decompressed image which was compressed by proposed design. Lossless compression mode of JPEG XR is used to test the implementation so recovered image is exactly same as original image.

http://dx.doi.org/10.5772/53100

(a) (b)

Figure 15. Figure (a) shows original image. Figure (b) shows decompressed image which was compressed by proposed LBT implementation.

5. Conclusion

In this chapter we have discussed the implementation of Lapped Biorthogonal Transform in LabVIEW for state of art image compression technique known as JPEG XR (ITU-T T.832 | ISO/IEC 29199-2). Such implementation can be used in PXI based high performance embedded controllers for image processing and compression. It also helps in research and efficient hardware implementation of JPEG-XR image compression. Moreover we also proposed an easily programmable, soft processor based design of LBT which requires less memory for processing that's makes this design suitable for low cost embedded devices.

Author details

Muhammad Riaz ur Rehman, Gulistan Raja and Ahmad Khalil Khan

Department of Electrical Engineering, University of Engineering and Technology, Taxila, Pakistan

References

[1] Wallace and G K. The JPEG still picture compression standard. IEEE Transactions on Consumer Electronics 1992; 38(1) xviii - xxxiv.

[2] Taubman and D S. JPEG2000: standard for interactive imaging. Proceedings of the IEEE 2002; 90(8) 1336 – 1357.

[3] ITU-T JPEG XR image coding system – Image coding specification. ITU-T Recommendation T.832; 2009.

[4] Frederic Dufaux, Gary Sullivan and Touradj. Ebrahimi. The JPEG XR Image Coding Standard. IEEE Signal Processing Magazine 2009; 26(6) 195-199.

[5] Malvar and H S.The LOT: transform coding without blocking effects. IEEE Transactions on Acoustics, Speech and Signal Processing 1989; 37(4) 553 – 559.

[6] Malvar and H S.Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts. IEEE Transactions on Signal Processing 1998; 46(4) 1043 – 1053.

[7] J Z Xu, F Wu, J Liang and W Zhang. Directional Lapped Transforms for Image Coding. IEEE Transactions on Image Processing 2010; 19(1) 85-97.

[8] Maalouf, A Larabi and M C. Low-complexity enhanced lapped transform for image coding in JPEG XR / HD photo. In: 16th IEEE International Conference on Image Processing (ICIP), 7-10 Nov. 2009, 5 – 8.

[9] M R Rehman and G Raja. A Processor Based Implementation of Lapped Biorthogonal Transform for JPEG XR Compression on FPGA. The Nucleus 2012; 49(3)

[10] Ching Yen Chien, Sheng Chieh, Huang, Chia Ho Pan, Ce Min Fang and Liang-Gee Chen. Pipelined arithmetic encoder design for lossless JPEG XR encoder. In: 13th IEEE International Symposium on Consumer Electronics, 25-28 May. 2009, 144 – 147.

[11] Groder, S H Hsu and K W. Design methodolgy for HD Photo compression algorithm targeting a FPGA. In: IEEE International SOC Conference, 17-20 Sept. 2008, 105 – 108.

[12] Ching Yen Chien, Sheng Chieh Huang, Shih-Hsiang Lin, Yu-Chieh Huang, Yi-Cheng Chen, Lei-Chun Chou, Tzu-Der Chuang, Yu-Wei Chang, Chia-Ho Pan and Liang-Gee Chen. A 100 MHz 1920×1080 HD-Photo 20 frames/sec JPEG XR encoder design.In: 15th IEEE International Conference on Image Processing (ICIP), 12-15 Oct. 2008, 1384 – 1387.

[13] Chia Ho Pan, Ching Yen Chien, Wei Min Chao, Sheng Chieh Huang and Liang Gee Chen. Architecture Design of Full HD JPEG XR Encoder for Digital Photography Applications. IEEE Transactions on Consumer Electronics 2008; 54(3) 963 – 971.