An Efficient Implementation of SPIHT Algorithm on a Reconfigurable Hardware

Ursila Khan, Arshad Aziz and Valiuddin Abbas
National University of Sciences and Technology (NUST), H-12, Islamabad, Pakistan
Pakistan Navy Engineering College, Karachi-75350, Pakistan
ursila.khan@pnec.edu.pk, arshad@nust.edu.pk, v_uddin@.pnec.edu.pk

Abstract— **This paper presents the implementation of the Set Partitioning in Hierarchical Trees (SPIHT) Image Compression Algorithm on reconfigurable platform. For flexibility of design and to achieve optimized results, we have combined the high-level utility of MATLAB with the flexibility and optimization of FPGA to implement this wavelet-based image compression coder on a sample 16 x 16 image. The initial and final stages of the design have been implemented on MATLAB while the coding and decoding being computationally intensive, are offloaded from the main process to the FPGA.**
.

I. INTRODUCTION

Over the last decade, the growing need of data compression has been felt in almost every aspect of our lives. With the ever growing technology, there has been a requirement of more and more efficient compression techniques. Multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth and for this reason, they are compressed using lossy or lossless compression techniques depending upon the area of application. [1]

We have chosen the SPIHT algorithm which is known to be implemented in Satellite Imaging, terrain imaging and also in bio-imaging. The algorithm works in conjunction with discrete wavelet transform and offers lossy but optimized compression results.

The structure of the paper is as follows. In Section 2, we have discussed the preliminaries and basic working of SPIHT algorithm. In Section 3 we have described the basic characteristics of a reconfigurable platform. Section 4 encompasses the software portion of our work, with explanation of DWT on MATLAB. In Section 5 we have described our design architecture and the coding and decoding SPIHT engines on FPGA. This also includes the integration of software and hardware platforms to achieve optimized results. In Section 6 we have briefly mentioned the implementation of our hardware design and tabulated the results. Section 7 pertains to the conclusions that we have arrived at, after combining the software and hardware computations. This is followed by future work in Section 8.

II. BASIC SPIHT ALGORITHM

SPIHT is an extension of the Embedded Zero Tree Algorithm (EZW) and was developed by Amir Said and William Pearlman in 1996. [2] It has been known to give significantly impressive results in image compression as compared to other techniques. The use of wavelets for satellite imaging, medical imaging has been advocated and its properties utilized in conjunction with SPIHT.

The SPIHT algorithm offers significantly improved quality over other image compression techniques such as vector quantization, JPEG and wavelets combined with quantization. [3] It offers characteristics such as: [2]

- Good image quality with a high PSNR
- Optimized for progressive image transmission
- Fast coding and decoding
- Can be used for lossless compression
- Can be efficiently combined with error protection

The SPIHT Algorithm works in two phases: First the wavelet transform of the input image is computed and then the wavelet coefficients are transmitted to the SPIHT coding engine.
After the discrete wavelet transform of the image has been computed, SPIHT divides the wavelets in to spatial orientation trees. Each node in the tree corresponds to an individual pixel. Each pixel in the transformed image is coded on basis of its significance by comparing with a threshold value at each level. If the value of a pixel or any of its offspring is below the threshold, we can conclude that all of its descendants are insignificant at that level and need not be passed.
After each pass, the threshold is divided by two and the algorithm proceeds further. This way information about the most significant bits of the wavelet coefficients will always precede information on lower-order significant bits, which is referred to as bit-plane ordering. [3]

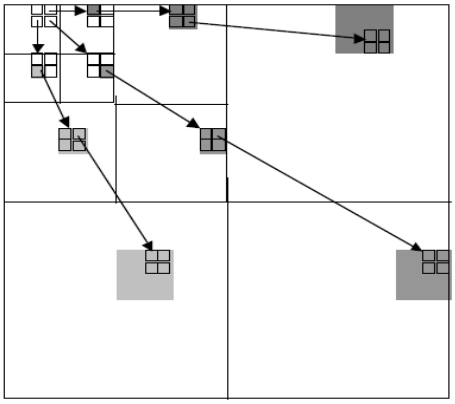

Figure 1: Spatial-Orientation Trees [2]

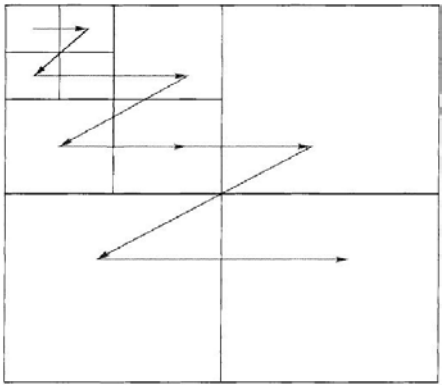

Figure 2: Scan order for coding the wavelet coefficients [4]

Fig 1 shows the root nodes with arrow pointing to the offspring of the nodes at each scale.By observing the above figure we can see that every node in the tree corresponds to an individual pixel. The descendants of a pixel are the four pixels in the same spatial location of the same sub band at the next finer scale of the wavelet. [3]
The wavelet coefficients are scanned in a zigzag pattern as shown in the figure. By following this manner of scan, an insignificant root automatically shows that its offspring are also insignificant. Fig 2 depicts this pattern.

As the algorithm proceeds, data are manipulated between three lists namely, the list of insignificant pixels (LIP), the list of insignificant sets (LIS) and the list of significant pixels (LSP). The LIP contains the pixels which on comparison with the threshold fall below it and are considered insignificant during that pass.
LIS contains sets of wavelet coefficients which are defined by tree structures, and which had been found to have magnitude smaller than a threshold (are insignificant). The sets exclude the coefficient corresponding to the tree or all sub-tree roots, and have at least four elements. [4] LSP contains the pixels which have values higher than the threshold and can be passed to the decoder during the current pass. Fig 3 shows how data are processed between the three lists.

III. FIELD-PROGRAMMABLE GATE ARRAY

A reconfigurable platform (FPGA) has been chosen due to its immense flexibility in terms of coding and memory allocation and manipulation. Our algorithm is computationally intensive and includes frequent manipulation of data between registers. For these reasons, the FPGA was chosen because it offers significant potential for the efficient implementation of a wide range of computationally intensive signal and image processing algorithms. [5]

Reconfigurable hardware also offers the advantage of look-up tables, which is used to replace a runtime computation with a simpler array indexing operation. The savings in terms of processing time can be significant, since retrieving a value from memory is often faster than undergoing an 'expensive' computation or input/output operation.[7] The FPGA also offers an additional advantage of its built-in hardware blocks such as the Block RAM, DCM modules etc, which we can utilize for better performance.

IV. DISCRETE WAVELET TRANSFORM

One of the most popular applications of wavelets has been to image compression. [4] Over the past several years, the wavelet transform has gained widespread acceptance in signal processing in general and in image compression research in particular. Wavelet based coding schemes have been known to give better performance in comparison to the schemes that use DCT. Also, wavelet-based coding is more robust under transmission and decoding errors, and also facilitates progressive transmission of images. [1]

In most image compression schemes, the Discrete Wavelet Transform (DWT) architecture is selected for exploiting the correlation among the image pixels and conveniently ignoring redundancies.

V. OUR DESIGN ARCHITECTURE

Our design architecture includes combination of software and hardware to implement the SPIHT algorithm. Since it's a wavelet-based coder, we computed the wavelet coefficients of our sample image on MATLAB and then used FPGA for coding and decoding the wavelets. After decoding, we reverted once again to MATLAB for the inverse wavelet transform of the coefficients to recover the original image in compressed form. We have explained the software and hardware implementation of our work separately.

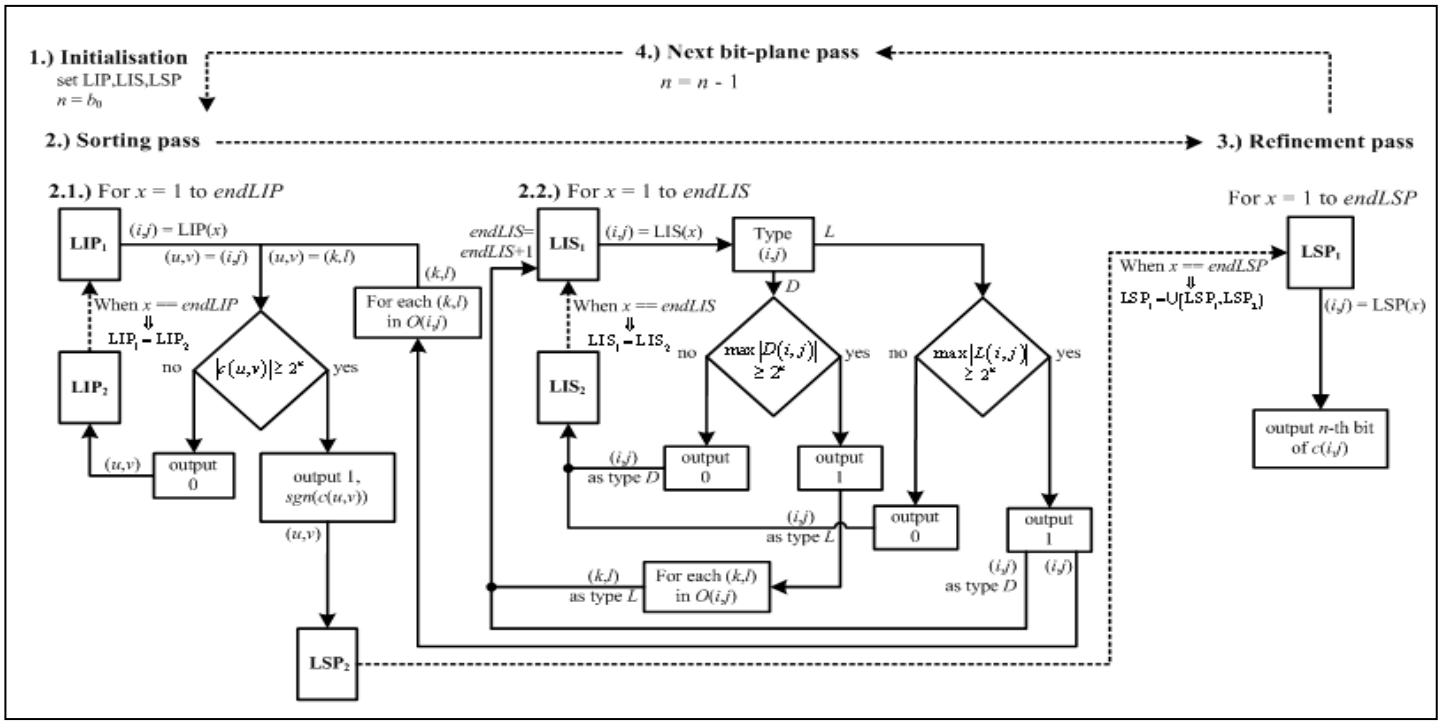

Figure 3: SPIHT Flowchart [8]

In our software implementation, we have discussed how we used the build-in toolbox of MATLAB to generate visual image decomposition. In the hardware implementation portion, we have explained the FPGA encoder and decoder working with flow charts. The charts explain the different modules of our coding. Fig 4 shows the workflow of our design.

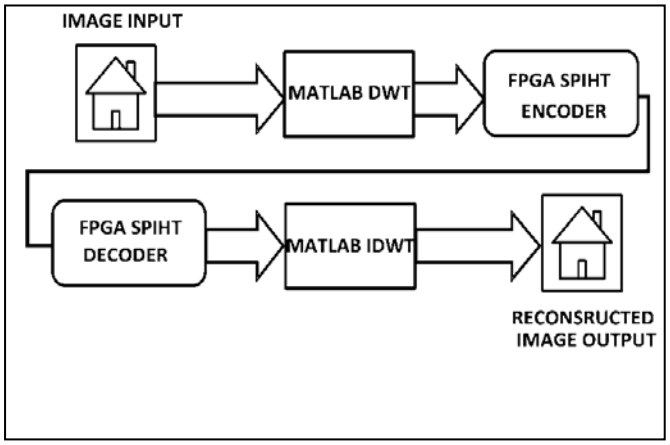

Figure 4: Design Workflow

A. DWT ON MATLAB

In our design, the wavelet coefficients of our image are calculated using MATLAB which are then transferred to the SPIHT Encoder. The encoder has been designed on Verilog to be used on FPGA. Similarly, the decoder has also been coded on Verilog for FPGA implementation. After decoding, the reconstructed wavelets are obtained, which are again given to MATLAB for the computation of the inverse wavelet transform. After the inverse dwt, the reconstructed image is retrieved.

We have used the Daubechies wavelet filter [9] to decompose our image at 4 levels. After computing the transform, the pixel coefficients are input to the FPGA encoder. Once the encoder has coded the pixels and the decoder has decoded the coefficients to retrieve the wavelets, an inverse wavelet transform is done to recover the image.

In our work we have decomposed a sample image using the wavelet toolbox from MATLAB [10], and shown the axes on which it has been decomposed. Also, we have shown the reconstructed image which has been obtained using the toolbox. This image serves as the target image which we intend to obtain using FPGA coding. The figures below show (Fig 5a) original sample image, (Fig 5b) its 4-level decomposition, (Fig 5c) axes of the image and (Fig 5d) target image reconstruction.

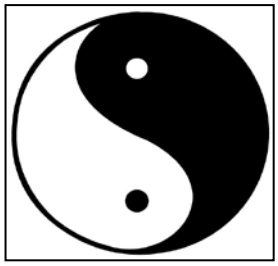

Figure 5a: Original Image

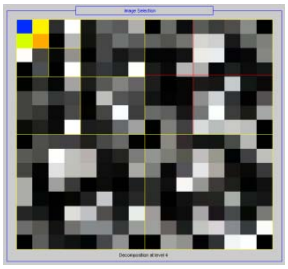

Figure 5b: Decomposition

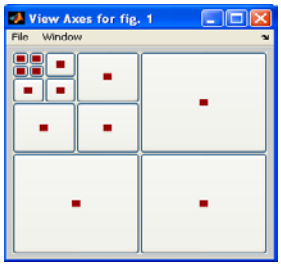

Figure 5c: Image Axes [9]

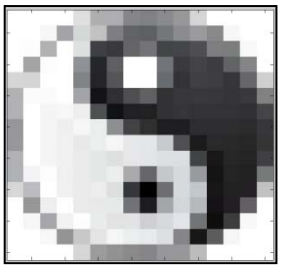

Figure 5d: Reconstruction

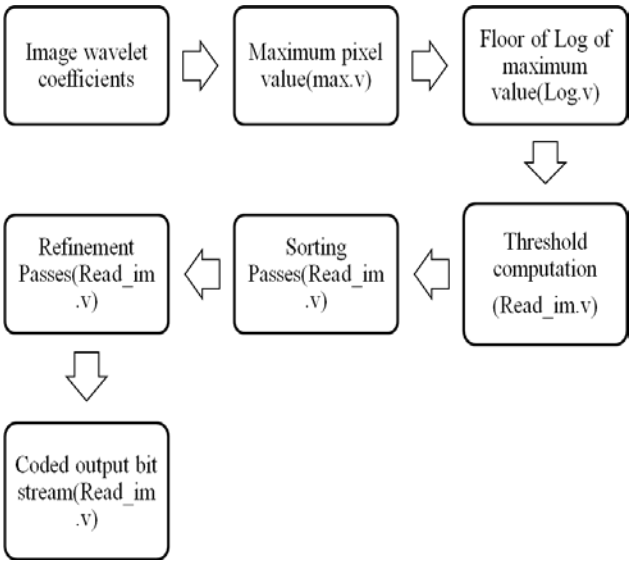

Figure 6: FPGA Encoder

B. SPIHT CODING AND DECODING ON FPGA

We have decomposed our image on four levels and coded it through five sorting passes. The number of sorting passes is continued till there remain no more sets in the LIS. Initially, our encoder takes the pixel coefficients in text files and computes the maximum magnitude pixel value in the file max.v as shown in the flow diagram Fig 6.This maximum value is then input to the Log.v file where its log base 2 and floor are calculated. This floor value is then used to compute the threshold, which is the base for all the comparisons in the sorting and refinement passes of our algorithm. After each pass on individual levels, a bit-stream is output which shall be used by the decoder to recover the image level by level. We have continued our algorithm for six Sorting Passes, till there remain no more sets in LIS. Fig 6 shows the work flow of our FPGA encoder.

At the decoder end, exactly the same pattern as the encoder is observed. The coded bit-stream is compared with the pixel locations and the values are reconstructed using a two-bit quantizer. The output bit-stream from each level of the encoder is used to reconstruct the wavelets at each level. After the entire image matrix is decoded, it is used to compute the inverse DWT by using MATLAB. The decoder working is shown in Fig 7.

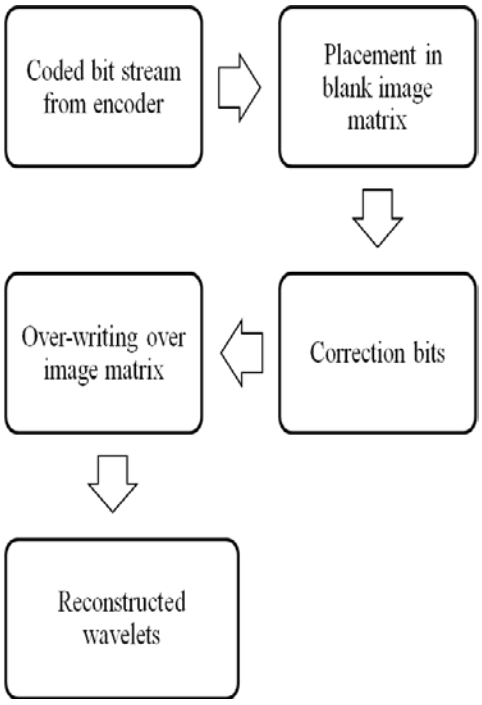

Figure 7: FPGA Decoder

C. INTEGRATION

We have computed the Discrete Wavelet Transform of our sample image on MATLAB. This is because image decomposition on a high-level language was considered a more convenient option. Also, decomposition on Verilog HDL requires very lengthy coding. This way, we employed MATLAB for the quick and easy wavelet computations. However, we chose FPGA for the coding process because it allows ease of rotating data within the memory arrays and registers and is a flexible option for algorithms which involve extensive computations and signal processes.

VI. IMPLEMENTATION RESULTS

We have successfully synthesized the first module of our FPGA encoder. The first module takes the wavelet coefficients of the image and computes the maximum value coefficient. This maximum magnitude coefficient is further used to compute threshold. Table 1 below gives the Device Utilizations and Timing details of the above mentioned module.
The remaining modules our work have been successfully checked for coding and are giving accurate results, however synthesis and implementation of design are currently in progress.

TABLE I.

IMPLEMENTATION TOOLS AND RESULTS

Target FPGA Device	xc3s500e-5-vq100		
Module Name	Max_value.v		
Device Utilization	Used	Available	Utilization
Number of Slices	35	960	3 %
Number of IOs	65		
Number of 4 input LUTs	67	1920	3 %
Number of bonded IOBs	65	66	98%
IOB Flip Flops	32		
Number of GCLKs	1	24	4 %
Timing Report			
Speed Grade	-5		
Maximum Period	No path found		
Minimum input arrival time before clock	1.731ns		
Maximum output required time after clock	11.752ns		
Maximum combinational path delay	13.581ns		

VII. CONCLUSION

We have successfully accomplished the software and hardware integration to implement the SPIHT algorithm on a sample 16 x 16 image. The reconstruction of our sample image is still under process and will require further work.

VIII. FUTURE WORK

In the future, this work can further be optimized to carry out all the phases on Verilog HDL. The Verilog HDL code can be optimized to achieve the desired results in lesser time and fewer storage elements. Also, the newer FPGA families being introduced in the market offer better performance and more storage elements to enhance performance and yield better results.

REFERENCES

[1] Subhasis Saha-"Image Compression - From DCT to Wavelets: A Review", Crossroads, Volume 6 , Issue (March2000) Pages: 12 - 21 Year of Publication: 2000 ,ISSN:1528-4972

[2] A. Said, W. A. Pearlman, "SPIHT Image Compression: Properties of the Method" http://www.cipr.rpi.edu/research/SPIHT/spiht1.html

[3] Thomas W. Fry and Scott Hauck, "SPIHT Image Compression on FPGAs", Circuits and Systems for Video Technology, IEEE Transactions on , Vol 15 Issue: 9, pp 1138 – 1147, Sept. 2005

[4] Amir Said and William Pearlman "Example of Application for Image Compression" http://www.cipr.rpi.edu/~pearlman/papers/ex_spiht-ezw.pdf

[5] Miriam Leeser, Srdjan Coric, Eric Miller, Haiqian Yu, Marc Trepanier, "Parallel-Beam Backprojection: an FPGA Implementation Optimized for Medical Imaging", The Journal of VLSI Signal Processing Volume 39, Number 3, 295-311, DOI: 10.1007/s11265-005-4846-5

[6] Khalid Sayood, "Introduction to Data Compression", Academic Press, Second Edition, Published 1996, pp 500-512

[7] Ian Kuon, Russell Tessier & Jonathan Rose- "FPGA ARCHITECTURE Survey and Challenges", Foundation and Trends in Electronic Design Automation, Volume 2 No 2, 2007, pp 135-253

[8] Nikola Sprljan, MATLAB Image and Video Compression Depot, SPIHT Algorithm Flowchart, http://sprljan.com/nikola/matlab/ztree.html

[9] Ingrid Daubechies, "Ten Lectures on Wavelets", Society for Industrial and Applied Mathematics**,**Year of Publication: 1992, ISBN:0-89871-274-2

[10] Wavelet Toolbox, MATLAB®, MathWorks, 2009.