VLSI ARCHITECTURE FOR LOW POWER VARIABLE LENGTH ENCODING AND DECODING FOR IMAGE PROCESSING APPLICATIONS

Vijaya Prakash. A.M[1] & K.S. Gurumurthy [2]
[1] Research Scholar, Dr. MGR University, Chennai, Faculty Department of ECE, BIT, Bangalore, India.
[2] Professor, Department of ECE, UVCE, Bangalore, India.

ABSTRACT
The image data compression has been an active research area for image processing over the last decade [1] and has been used in a variety of applications. This paper investigates the implementation of Low Power VLSI architecture for image compression, which uses Variable Length Coding method to compress JPEG signals [1]. The architecture is proposed for the quantized DCT output [5]. The proposed architecture consists of three optimized blocks, viz, Zigzag scanning, Run-length coding and Huffman coding [17]. In the proposed architecture, Zigzag scanner uses two RAM memories in parallel to make the scanning faster. The Run-length coder in the architecture, counts the number of intermediate zeros in between the successive non-zero DCT coefficients unlike the traditional run-length coder which counts the repeating string of coefficients to compress data [20]. The complexity of the Huffman coder is reduced by making use of a lookup table formed by arranging the {run, value} combinations in the order of decreasing probabilities with associated variable length codes [14]. The VLSI architecture of the design is implemented [12] using Verilog HDL with Low Power approches . The proposed hardware architecture for image compression was synthesized using RTL complier and it was mapped using 90nm standard cells. The Simulation is done using Modelsim. The synthesis is done using RTL compiler from CADENCE. The back end design like Layout is done using IC Compiler. Power consumptions of variable length encoder and decoder are limited to 0.798mW and 0.884mW with minimum area. The Experimental results confirms that 53% power saving is achieved in the dynamic power of huffman decoding [6] by including the lookup table approach and also a 27% of power saving is achieved in the RL-Huffman encoder [8].

KEYWORDS: *Variable Length Encoding (VLE), Variable Length Decoding (VLD), Joint Photographic Expert Group (JPEG), Image Compression, Low Power Design, Very Large Scale Integration (VLSI).*

I. INTRODUCTION

Image data compression refers [4] to a process in which the amount of data used to represent image is reduced to meet a bit rate requirement (below or at most equal to the maximum available bit rate) while the quality of the reconstructed image satisfies a requirement for a certain application and the complexity of computation involved is affordable for the application [18]. The image compression can improve the performance of the digital systems by reducing the cost of image [22] storage and the transmission time of an image on a bandwidth limited channel, without significant reduction in the image quality [15].

This paper investigates the implementation of Low Power VLSI architecture for image compression [8] which uses variable length coding method for image data compression, which could be then used for practical image coding systems to compress JPEG signals [1].

Variable length coding [2] that maps input source data on to code words with variable length is an efficient method to minimize average code length. Compression is achieved by assigning short code words to input symbols of high probability and long code words to those of low probability [14]. Variable length coding can be successfully used to relax the bit-rate requirements [21] and storage spaces for many multimedia compression systems. For example, a variable length coder (VLC) employed in MPEG-2 along with the discrete cosine transform (DCT) results [16] in very good compression efficiency.

Since early studies have focused only on high throughput variable length coders [6], low-power variable length coders have not received much attention. This trend is rapidly changing as the target of multimedia systems is moving toward portable applications like laptops, mobiles and iPods etc [15]. These systems highly demand low-power operations, and, thus require low power functional units.

The remaining paper is organized as follows, Section 2 explains about the practical needs, principles and types of image compression, Section 3 explains the Variable Length Encoding Process. Section 4 describes Variable Length Decoding Process, Section 5 includes the Interpretation of Results. Section 6 finally concludes the paper.

II. IMAGE COMPRESSION

A. Practical Needs for Image Compression

The need for image compression becomes apparent when number of bits per image is computed resulting from typical sampling rates and quantization methods [4]. For example, the amount of storage required for given images is (i) a low resolution, TV quality, color video image which has 512×512 pixels/color,8 bits/pixel, and 3 colors approximately consists of 6×10^6 bits; (ii) A 24×36 mm negative photograph scanned at 12×10^{-6}mm: 3000×2000 pixels/color, 8 bits/pixel, and 3 colors nearly contains 144×10^6 bits; (iii) a 14×17 inch radiograph scanned at 70×10^{-6}mm: 5000×6000 pixels, 12 bits/pixel nearly contains 360×10^6 bits. Thus storage of even a few images could cause a problem. As another example of the need for image compression [15], consider the transmission of low resolution 512×512×8 bits/pixel×3- color video image over telephone lines. Using a 96000 bauds (bits/sec) modem, the transmission would take approximately 11 minutes for just a single image [22], which is unacceptable for most applications.

B. Principles behind Compression

Number of bits required to represent the information in an image can be minimized by removing the redundancy present in it. There are three types of redundancies.

1. Spatial redundancy, which is due to the correlation or dependency between neighboring pixel values.
2. Spectral redundancy, which is due to the correlation between different color planes or spectral bands.
3. Temporal redundancy, which is present because of correlation between different frames in images.

Image compression research [18] aims to reduce the number of bits required to represent an image by removing the spatial and spectral redundancies as much as possible [22].

C. Types of Image Compression

Compression can be divided into two categories [1], Lossless and Lossy compression. In lossless compression schemes, the reconstructed image after compression is numerically identical to the original image. However lossless compression can only achieve a modest amount of compression. Lossless compression is preferred for archival purposes like medical imaging [22], technical drawings, clip art or comics. This is because lossy compression methods, especially when used at low bit rates [21], introduce compression artifacts. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards redundant information. However the lossy schemes are capable of achieving much higher compression. Lossy methods are especially suitable for natural images such as photos in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to

achieve a substantial reduction in bit rate. The lossy compression is produces imperceptible differences can be called visually lossless.

D. Image Compression using Discrete Cosine Transform

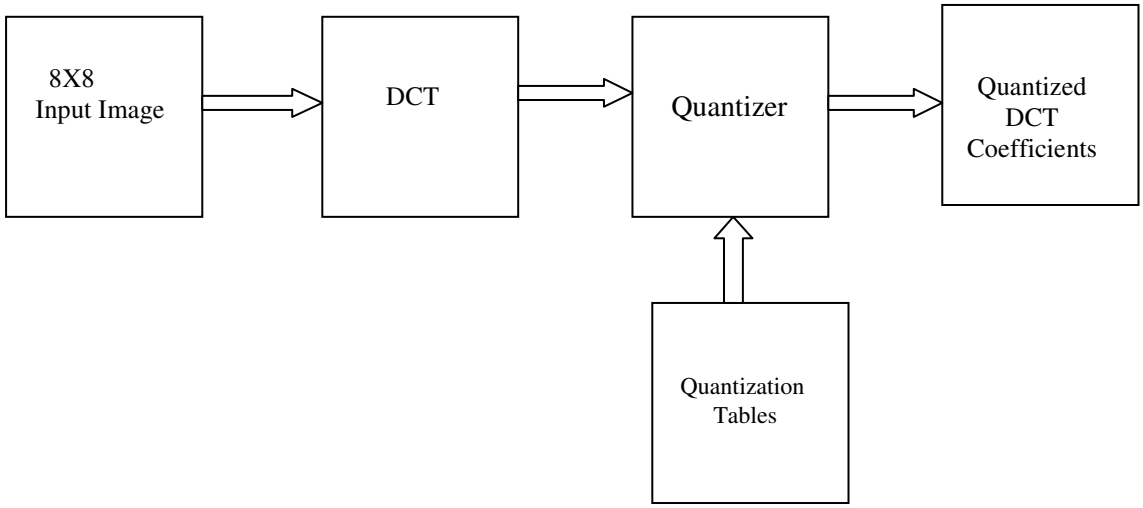

Figure1. Block diagram of DCT Process

In the DCT process input image is divided into non- overlapping blocks of 8×8 pixels [4], and input to the baseline encoder. The pixel values are converted from unsigned integer format to signed integer format, and DCT computation is performed on each block. DCT transforms the pixel data into a block of spatial frequencies that are called the DCT coefficients. Since the pixels in the 8×8 neighbourhood typically have small variations in gray levels, the outputs of the DCT will result in most of the block energy being stored in the lower spatial frequencies [15]. On the other hand, the higher frequencies will have values equal or close to zero and hence, can be ignored during encoding [9] without significantly affecting the image quality. The selection of frequencies based on which frequencies are most important [15] and which ones are less important can affect the quality of the final image.

The selection of quantization values is critical since it affects both the compression efficiency [4], and the reconstructed image quality. High frequency coefficients have small magnitude for typical video data, which usually does not change dramatically between neighbouring pixels. Additionally, the human eye is not as sensitive to high frequencies as to low frequencies [5]. It is difficult for the human eye to discern changes in intensity or colors that occur between successive pixels. The human eye tends to blur these rapid changes into an average hue and intensity. However, gradual changes over the 8 pixels in a block are much more discernible than rapid changes. When the DCT is used for compression purposes, the quantizer unit attempts to force the insignificant high frequency coefficients to zero while retaining the important low frequency coefficients. The 2-D DCT transforms an 8×8 block of spatial data samples into an 8×8 block of spatial frequency components [15]. These DCT coefficients are then used as input to the Variable Length Encoder which will further compress the image data [9]. The compressed image data can be decompressed using Variable Length Decoder [10] and then IDCT transforms spatial frequency components back into the spatial domain [15] to successfully reconstruct the image.

III. VARIABLE LENGTH ENCODING

Variable Length Encoding (VLE) is the final lossless stage of the video compression unit.VLE is done to further compress the quantized image [13]. VLE consists of the following three steps:

- *Zigzag scanning*
- *Run Length Encoding (RLE), and*
- *Huffman coding.*

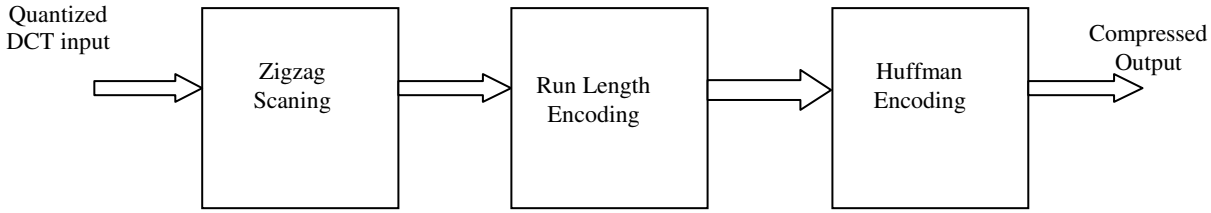

Figure 2.Variable Length Encoder

A. Zigzag Scanning

Figure 3. Block diagram of Zigzag Scanner

The quantized DCT coefficients obtained after applying the Discrete Cosine Transformation to 8×8 block of pixels they are fed as input to the Variable Length Encoder (VLE). These quantized DCT coefficients will have non-zero low frequency components in the top left corner of the 8×8 block and higher frequency components in the remaining places [17]. The higher frequency components approximate to zero after quantization. The low frequency DCT coefficients are more important than higher frequency DCT coefficients. Even if we ignore some of the higher frequency coefficients, we can successfully reconstruct the image from the low frequency coefficients only. The Zigzag Scanner block exploits this property [7]. In zigzag scanning, the quantized DCT coefficients are read out in a zigzag order, as shown in the figure 4. By arranging the coefficients in this manner, RLE and Huffman coding can be done to further compress the data. The scan puts the high-frequency components together. These components are usually zeroes.

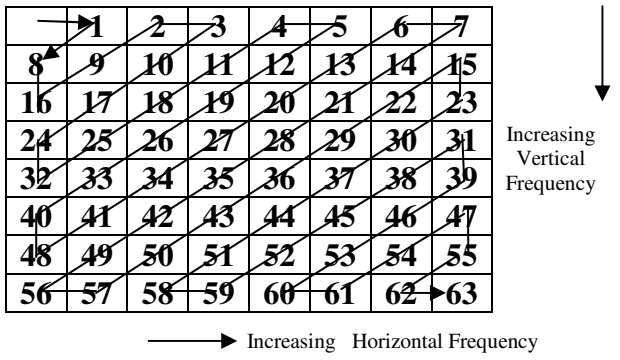

Figure 4. Zigzag Scan Order

Since the zigzag scanning requires that all the 64 DCT coefficients are available before scanning, we need to store the serially incoming DCT coefficients in a temporary memory. For each of the 64 DCT coefficients obtained for each 8×8 block of pixels we have to repeat this procedure. So at a time either scanning is performed or storing of incoming DCT coefficients is done. This will slow down the scanning process. So in order to overcome this problem and to make scanning faster, we propose a new architecture for zigzag scanner. In the proposed architecture, two RAM memories will be used in the zigzag scanner [17]. One of the two RAM memories will be busy in storing the serially incoming DCT coefficients while scanning is performed from the other RAM memory. So except for first 64

clock cycles i.e., until 64 DCT coefficients of first 8×8 blocks become available, the zigzag scanning and storing of serially incoming DCT coefficients is performed simultaneously [11]. So by using two RAM memories we will able to scan one DCT coefficient in each clock cycle except for first 64 clock cycles.

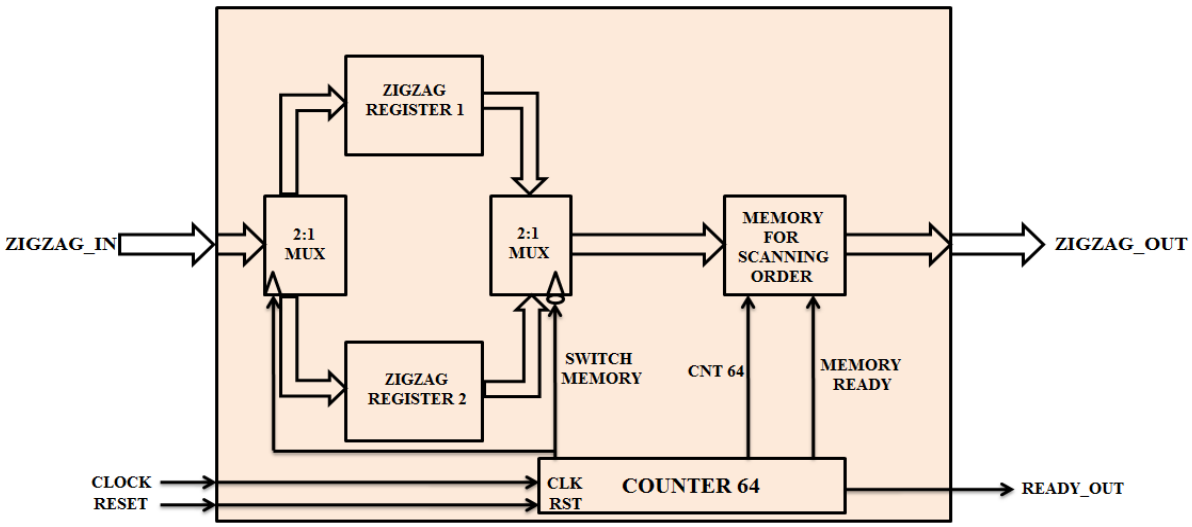

Figure 5. Internal Architecture of the Zigzag Scanner

B. Run Length Encoding (RLE)

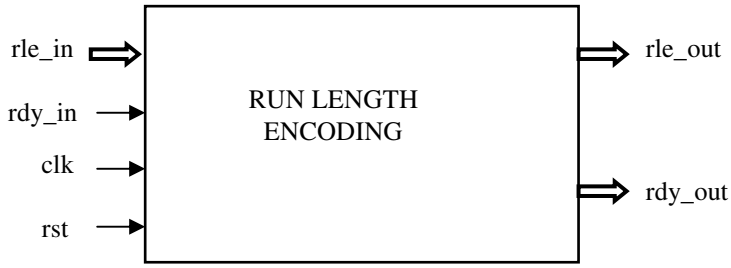

Figure 6. Block diagram of Run- Length Encoder

The quantized coefficients are read out in a zig-zag order from DC component to the highest frequency component. RLE is used to code the string of data from the zig-zag scanner. The conventional Run length encoder codes the coefficients in the quantized block into a run length (or number of occurrences) and a level or amplitude. For example, transmits four coefficients of value "10" as: {10,10,10,10}. By using RLE [8], the level is 10 and the run of a value of 10 is four. By using RLE, {4,10} is transmitted, reducing the amount of data transmitted. Typically, RLE [10] encodes a run of symbols into two bytes, a count and a symbol. By defining an 8 x 8 block without RLE, 64 coefficients are used. To further compress the data, many of the quantized coefficients in the 8 x 8 block are zero. Coding can be terminated when there are no more non-zero coefficients in the zig-zag sequence [9]. Using the "end-of-block" code terminates the coding.

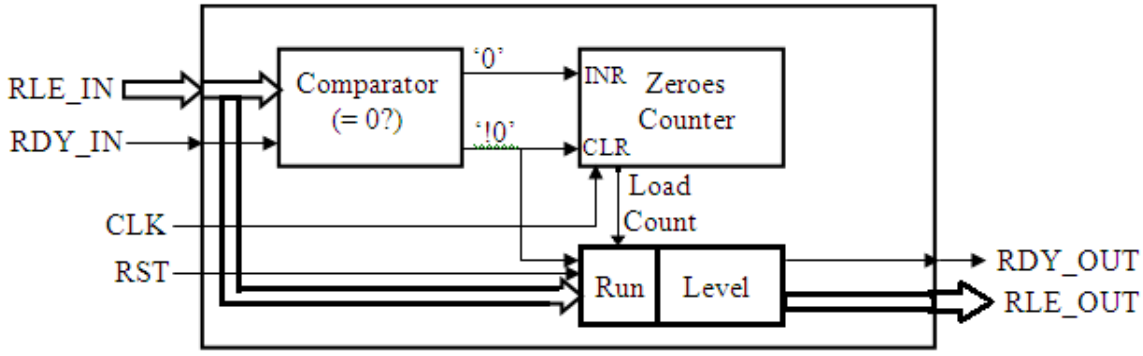

Figure 7. Internal architecture of run-length encoder.

But normally in a typical quantized DCT matrix the number of zeroes is more [5] compared to non-zero coefficients being repeated [4]. So in the proposed architecture for run-length encoder we exploit this property of more number of zeroes in the DCT matrix. In the proposed architecture the number of intermediate zeros in between non-zero DCT coefficients are counted [10] unlike the conventional run-length encoder where number of occurrences of repeated symbols are counted.

Conventional RLE	**Proposed RLE**
31	31
1,0	1,1
1,1	1,2
1,0	0,1
1,2	5,2
1,1	EOB
5,0	
1,1	
EOB	

Figure 8. Comparison between conventional and proposed RLE

C. Huffman encoding

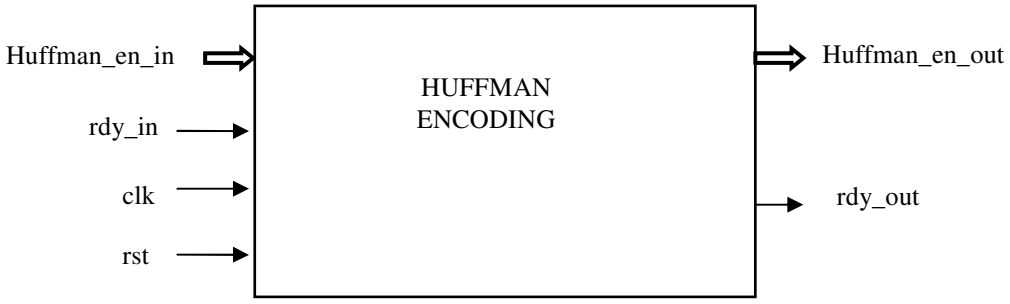

Figure 9. Block diagram of Huffman Encoder

Huffman coding is used to code values statistically according to their probability of occurrence [11]. Short code words are assigned to highly probable values and long code words to less probable values. The procedure for Huffman coding involves the pairing of run/value combinations. The input run/value combinations are written out in the order of decreasing probability. The run/value combination with the highest probability is written at the top, the least probability is written down last. The least two probabilities are then paired and added. A new probability list is then formed with one entry as the previously added pair. The least run/value combinations in the new list are then paired. This process is continued till the list consists of only one probability value. The values "0" and "1" are arbitrarily assigned to each element in each of the lists.

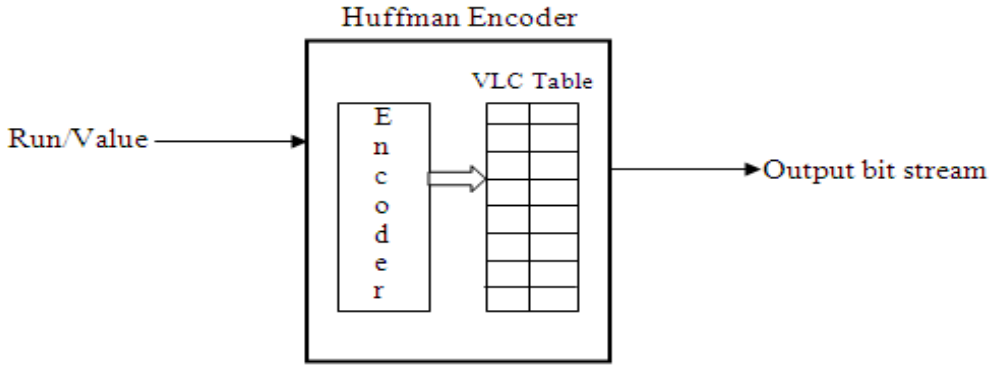

Figure 10. Internal architecture of Huffman encoder

In the proposed architecture, the Huffman encoding is done by making use of a lookup table [3]. The lookup table is formed by arranging the different run-value combinations in the order of their probabilities of occurrence with the corresponding variable length Huffman codes [6]. When the output of the run- length encoder in form of run-value combination is fed to the Huffman encoder, the run-value combination received will searched in the lookup table, when run-value combination is found its corresponding variable length Huffman code [14] will be sent to output. This approach of designing Huffman encoder not only simplifies the design but also results in less power consumption [6]. Since we are using lookup table approach, the only part of encoder corresponding to the current run-length combination will be active and other parts of the encoder will not be using any power. So turning off the inactive components of a circuit in the Huffman encoder, results in less power consumption.

IV. VARIABLE LENGTH DECODING

The variable length decoder is the first block on the decoder side. It decodes the variable length encoder output to yield the quantized DCT coefficients [10]. The variable length decoder consists of three major blocks, namely,

1. Huffman Decoding.
2. Run- length Decoding.
3. Zigzag Inverse Scanning.

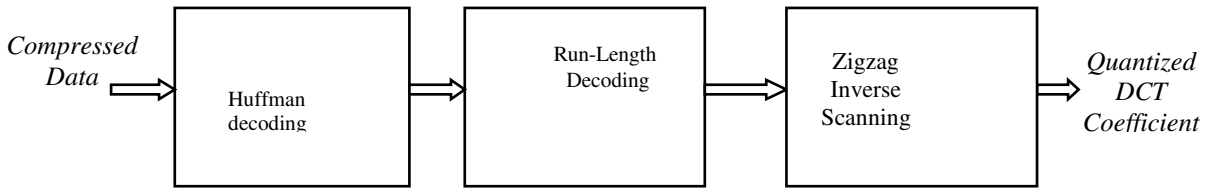

Figure11. Variable Length Decoder

A. Huffman Decoding

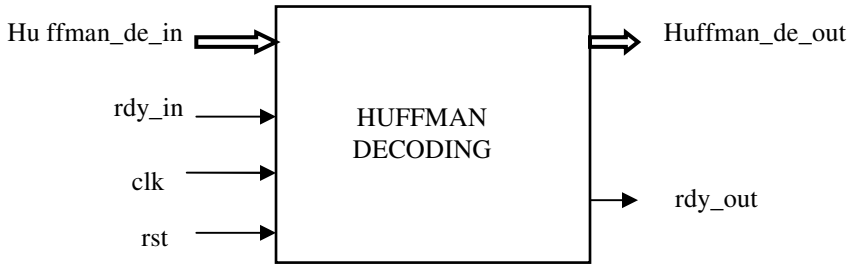

Figure 12. Block diagram of Huffman decoder

The huffman decoder forms the front end part of the variable length decoder. The internal architecture of the huffman decoder is same as the huffman encoder. The same VLC huffman coding table which was used in the huffman encoder is also used in the huffman decoder [17]. The input encoded data is taken and a search is done for the corresponding run/value combination in the VLC table. Once the corresponding run/value combination [14] is found, it is sent as output and huffman starts decoding next coming input.

The VLC huffman coding table which we are using in both the huffman encoder and the huffman decoder, reduces the complexity of the huffman decoder. It is not only reduces the complexity [7] but also reduces the dynamic power in the huffman decoder, since only the part of the circuit is active at a time.

B. FIFO

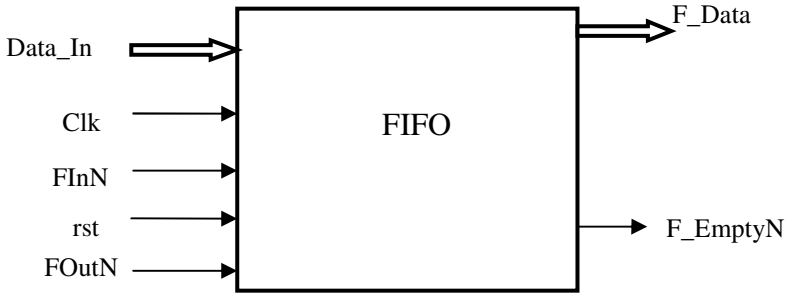

Figure 13. Block diagram of FIFO

The First In First Out (FIFO) also forms the part of the decoder part, The FIFO is used between the huffman decoder and the run-length decoder. The FIFO is used to match the operating speed between the huffman decoder and run-length decoder [11]. The huffman decoder sends a decoded output to the run-length decoder in the form of run/value combination. The run-length combination takes this as input and starts decoding [12]. Since here the run in the run/value combination represents the number of zeroes in between consecutive non-zero coefficients, the zero '0' is sent as output for next 'run' number of clock cycles. Until then the run-length decoder can't accept other run/value combination. And we know that the huffman decoder decodes one input to one run/value combination in every clock cycle. So huffman decoder can't be connected directly to run-length decoder. Otherwise the run-length decoder can't decode correctly. So to match the speed between the huffman decoder and run-length decoder, the FIFO is used. The output of the huffman decoder is stored onto the FIFO, the run-length decoder takes one decoded output of huffman decoder from the FIFO when it is finished with decoding of the present input to it. So after run-length decoder finishes decoding of the present input, it has to send a signal to the FIFO to feed it a new input. This signal is sent to the FOutN pin, which is read out pin of the FIFO. The FInN pin is used to write onto FIFO, the huffman decoder generates the signal for this while it has to write a new input onto the FIFO. So the FIFO acts as a synchronizing device between the huffman decoder [9] and the run-length decoder.

C. Run-Length Decoder

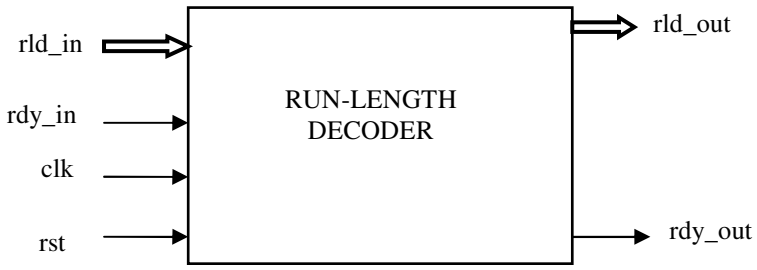

Figure14. Block diagram of Run- Length Decoder

The Run-length decoder forms the middle part of the variable length decoder [10]. It takes decoded output from huffman decoder through FIFO. When the huffman decoder decodes one input and stores the decoded output onto the FIFO, then the FIFO becomes non-empty (the condition when at least one element is stored on the FIFO). The FIFO then generates the signal F_EmptyN. This signal is used as rdy_in signal to the run-length decoder. So when huffman decoder decodes one input and stores it onto the FIFO[7], then a ready signal is generated to the run-length decoder to initiate the decoding process.

The run-length decoder takes the input in the form of a run/value combination, then separates run and value parts. The run here represents number of zeroes to output before sending out the non-zero level 'value' in the run/value combination. So for example if {5,2} is input to the run-length decoder then it sends 5 zeroes (i.e., 5 '0') before transmitting the non-zero level '2' to the output. Once the run-length decoder sends out a non-zero level, then it means that it is finished with the decoding of the present run/value combination, and it is ready for the next run/value combination. So for this it generates the rdy_out signal to the FIFO, to indicate that it has finished decoding of present input and ready for decoding the next run/value combination. This rdy_out is connected to the FOutN pin of the FIFO, which is read out pin of the FIFO[16]. Upon receiving this signal the FIFO sends out a new run/value combination to the run-length decoder, to initiate run-length decoding process for the new run/value combination.

D. Zigzag Inverse Scanner

Figure15. Block diagram of Zigzag Scanner

The zigzag inverse scanner forms the last stage in the variable length decoder [17]. The working and architecture, everything is similar to the zigzag scanner, except that the scanning order will be different. The zigzag inverse scanner gets the input from the run-length decoder, starts storing them in one of the two RAMs [13], until it receives all 64 coefficients. Once it receives all the 64 coefficients, it starts inverse scanning to decode back the original DCT coefficients. Meanwhile the incoming DCT coefficients are getting stored in another RAM [2]. Once scanning from one RAM is finished, it starts scanning from another RAM and meanwhile the incoming DCT coefficients gets stored in first RAM [5]. So this process is repeated until all the DCT coefficients are scanned. There will be delay of 64 clock cycles before the output appears. Once after that, for every clock cycle an element will scanned continuously.

V. INTERPRETATION OF RESULTS

1.Simulation Results

A. *Zigzag Scanning*

The simulation of zigzag scanning is done using the following test sequence.

31	0	1	0	0	0	0	0	1	2	0	0	….	0	0
0	1	2	3	4	5	6	7	8	9	10	11	62	63	

The figure 16 shows the simulation waveform of the zigzag encoding block for the above test sequence.

B. *Run Length Encoding*

The simulation of the run-length encoding block is done using the output sequence obtained in zigzag scanning process, which appears at the input of the run-length encoder as below.

31	0	1	0	2	1	0	0	0	0	0	2	……	0	0
0	1	8	16	9	2	3	10	17	24	32	25	62	63	

The figure 17 shows the simulation waveform of the run-length encoding block for the above test input sequence.

C. *Huffman Encoding*

31	1,1	1,2	0,1	5,2	EOB

The output of the run-length encoder is used as the test sequence to the Huffman encoder. The output of the run-length encoder will appear as below. The figure 18 shows the simulation waveform of the huffman encoding block for the above test input sequence.

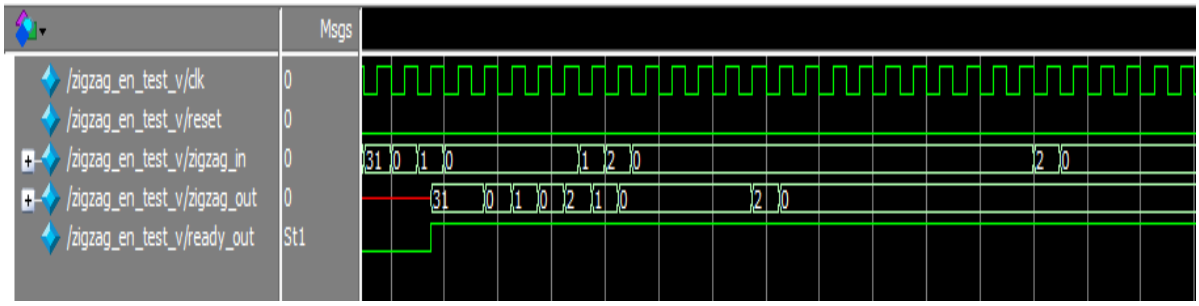

Figure16. Simulation waveform of Zigzag Scanning

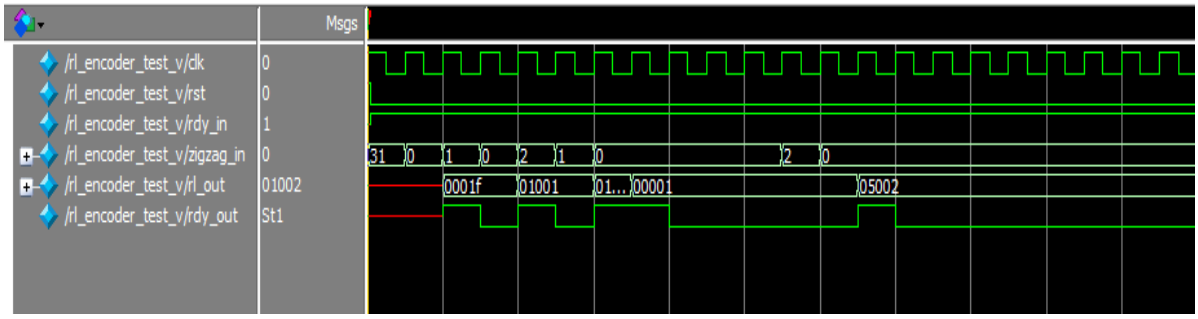

Figure17. Simulation waveform of Run-length Encoder

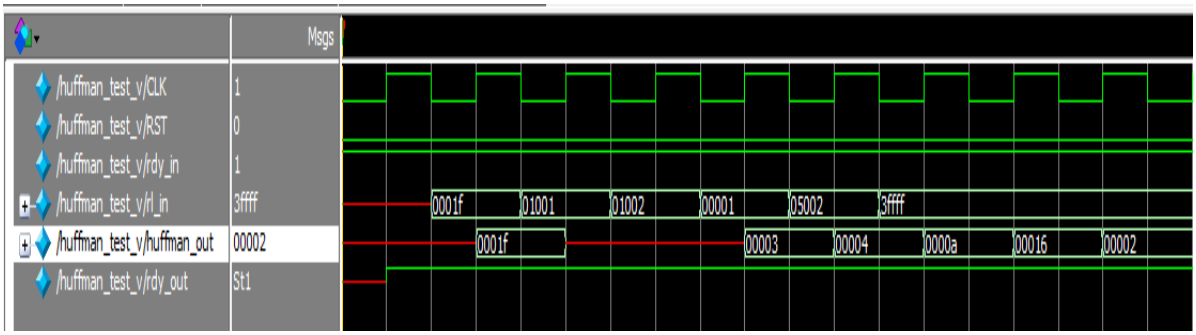

Figure18. Simulation waveform of Huffman Encoder

D. Huffman Decoding

The compressed data from the variable length encoder is fed to the Huffman decoding block. The output obtained from the Huffman encoding block is used as a test bench to the Huffman decoding block. The fig 19 shows the simulation waveform for the Huffman decoding block.

E. Run Length Decoding

After Huffman decoder decodes the compressed bitstream, the decoded input is fed to run-length decoder block. The fig 20 shows the simulation waveform of run-length decoder.

F. Zigzag Inverse Scanner

The output of the run-length decoder is given as input to the zigzag inverse scanner, which will output the quantized DCT coefficients. The fig 21 shows the simulation waveform of Zigzag Inverse Scanner.

2. Synthesis Results

Once all the behavioral verifications (simulation) are done, syntheses of blocks are performed. The Cadence RTL compiler is used for synthesis and design is implemented on 90nm standard cells. The layout is generated using VLSI Backend EDA tool. After synthesis of each blocks the area and power analysis are performed for each of the block used in the design. The Layout of the Run Length Encoder is shown in the figure 22.

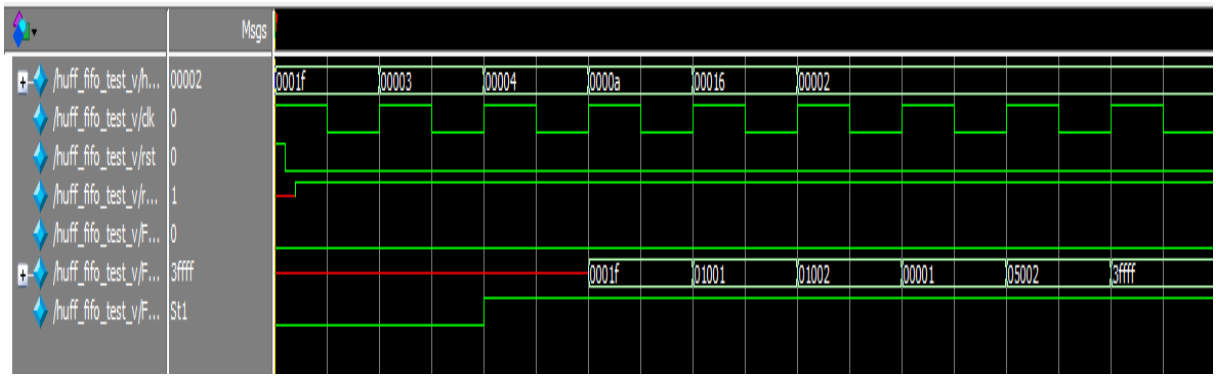

Figure19. Simulation waveform of Huffman Decoder

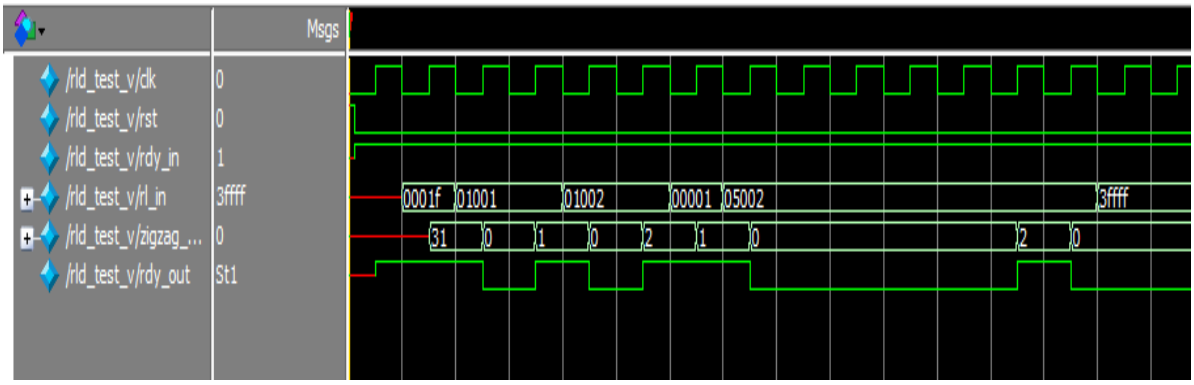

Figure20. Simulation waveform of Run-length Decoder

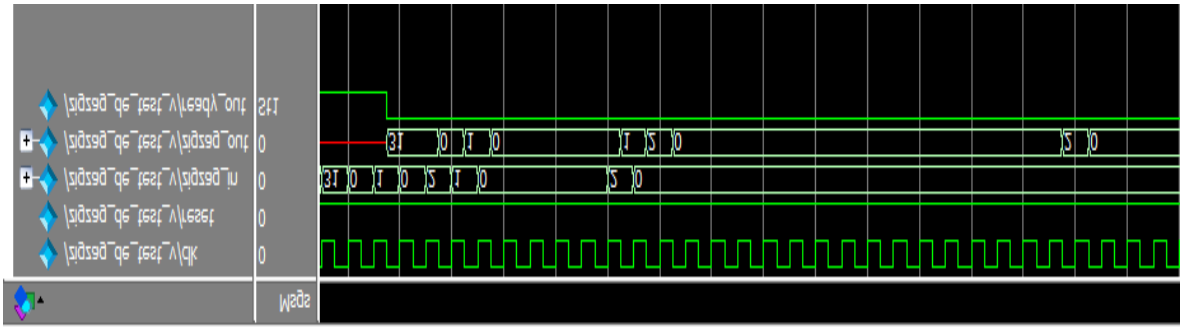

Figure21. Simulation waveform of Zig- Zag Inverse Scanner

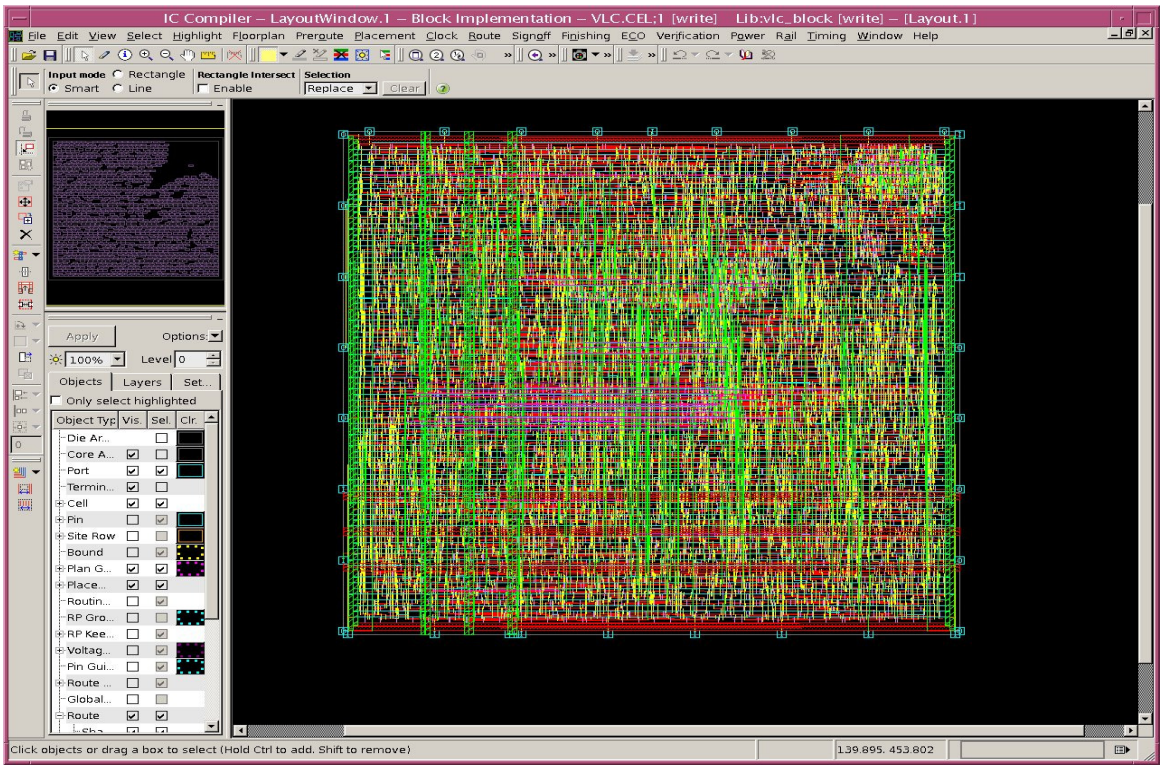

Figure22. Layout of Variable Length Encoder

The power and area characteristics of each of the blocks in the design are tabulated as shown in Table 1, and the characteristics are represented in the form of graph are also show in graph 1.

Table 1: Power & Area Characteristics

Designs \ Features	Power(in mW)	Area
Zigzag Scanner	0.7319	0.3349
Run Length Encoder	0.0208	0.0103
Huffman Encoder	0.0451	0.0171
Zigzag Inverse Scanner	0.7285	0.3359
un Length Decoder	0.1110	0.0071
Huffman Decoder	0.0451	0.0171
FIFO	0.2744	0.1287

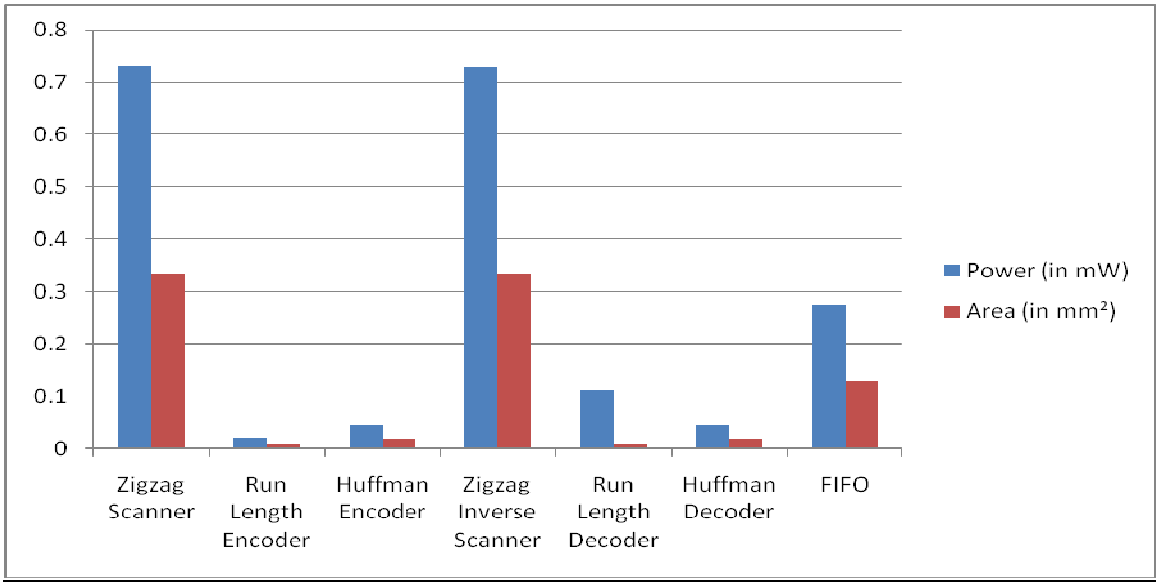

Figure 23. Representation of Power & Area Characteristics

3. Power Comparisons

The power comparison of the proposed Architecture is as shown below.

3.1 Power comparison of Huffman Decoder

Table 2: Power Comparison for Huffman decoders

Table Size	Huffman decoder type	Power (in µW)
100	Power Analysis of the Huffman Decoding Tree, by Jason McNeely [6]	95
	Proposed Architecture	45

3.2 Power comparison of RL- Huffman Encoder Combination

Table 3: Power Comparison for RL-Huffman Encoders

RL-Huffman Encoding Type	Power (in µW)
RL-Huffman Encoding for Test Compression and Power Reduction in Scan Applications [8]	90
Proposed Architecture	65.9

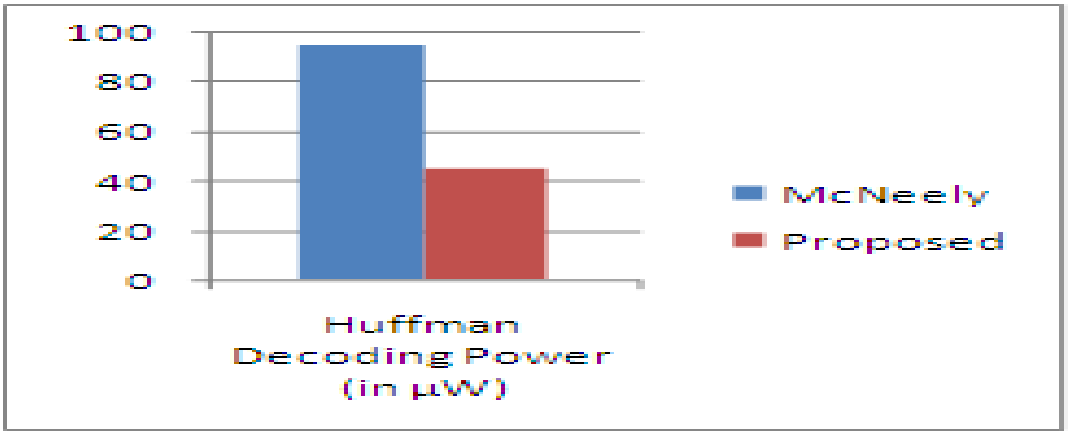

Figure 24. Power Comparison for Huffman decoders

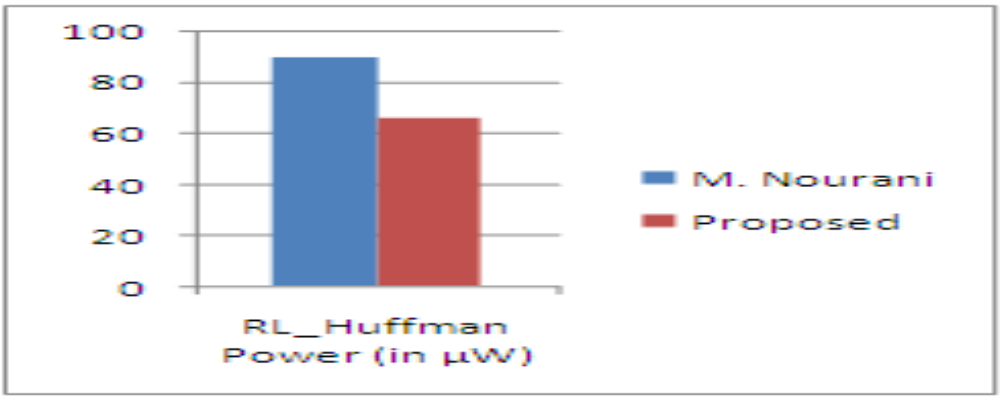

Figure 25. Power Comparison for RL-Huffman

3.3 Percentage of Power Saving

The percentage of power savings from proposed design are calculated and are tabulated as shown in following table 4.

Table 4: Percentage of Power Savings

Encoding Type	Comparison with	Percentage of Power Saving
Huffman decoder	Power Analysis of the Huffman Decoding Tree [6]	52.63%
RL-Huffman Encoder	RL-Huffman Encoding for Test Compression and Power Reduction in Scan Applications [8]	26.77%

VI. CONCLUSION

In this paper we described a Low Power Architecture for Variable Length Encoding and Variable Length Decoding for image processing applications. The designing and modeling of the all the blocks in the design is done by using synthesizable Verilog HDL with Low Power approach. The proposed architecture is synthesized using RTL complier and it is mapped using 90 nm standard cells. The Simulation of all the blocks in the design was done using Modelsim. A detailed analysis for power and area was done using RTL compiler from CADENCE. Power consumptions of variable length encoder and decoder are limited to 0.798mW and 0.884mW with minimum area. A 53% power saving is achieved in the dynamic power of huffman decoding [6] by including the lookup table approach and also a 27% of power saving is achieved in the RL-Huffman encoder [8].

REFERENCES

[1] G. Wallace, "*The JPEG still-image compression standard*," Commun. ACM, vol. 34, pp. 30–44, Apr. 1991.ACM, vol. 34, pp. 30–44, Apr. 1991.

[2] *Low Power Look-Up Tables for Huffman Decoding*, by Jason McNeely and Magdi Bayoumi, The Centre of Advanced Computer Studies, University of Louisiana, Lafayette.

[3] *Direct Huffman Coding and Decoding using the Table of Code- Lengths*, by Reza Hashemian, Senior Member, IEEE Northern Illinois University DeKalb, Illinois 60115, USA.

[4] *A Novel Fast DCT Coefficient Scan Architecture*, by Da An, Xin Tong, Bingqiang Zhu and Yun He, State Key Laboratory on Microwave and Digital Communications Tsinghua National Laboratory for Information Science and echnology Department of Electronic Engineering, Tsinghua University, **B**eijing 100084, China.

[5] A Novel VLSI Architecture for Image Compression using DCT and Quantization. By Vijaya Prakash A M and K. S Gurumurthy, IJCSNS Vol .10 No.9 September 2010.

[6] "*Power Analysis of the Huffman Decoding Tree*", Jason McNeely, Yasser Ismail, Magdy A. Bayoumi and Peiyi Zaho. The Center for Advanced Computer Studies, University of Louisiana at Lafayette.IEEE Conference on Image Processing (ICIP) California USA October 2008.

[7] D.A. Huffman, "*A method for construction of minimum-redundancy codes*", Proc. IRE, Vol. 40,pp. 1098-1101, Sept. 1952.

[8] "*RL-Huffman Encoding for Test Compression and Power Reduction in Scan Applications*". Mehrdad Nourani and Mohammed H. Tehranipour Center for Integrated Circuits and Systems, The University of Texas at Dallas.ACM Transactions on Design Automation of Electronic Systems vol.10.No1. Jan 2005.

[9] *Joint Optimization of Run-Length Coding, Huffman Coding, and Quantization Table with Complete Baseline JPEG Decoder Compatibility*, by En-hui Yang, Fellow, IEEE, and Longji Wang, Member, IEEE.

[10] *A Low-Power Variable Length Decoder for MPEG- 2 Based on Successive Decoding of Short Codewords* Sung-Won Lee, Student Member, IEEE, and In-Cheol Park, Senior Member, IEEE.

[11] *A Fast Parallel Huffman Decoder for FPGA Implementati* ACTA TECHNICA NAPOCENSIS Electronics and Telecommunications Volume 49, Number 1, 2008.

[12] *'JPEG Architecture and Implementation Issues'*, by Dr. Sri Krishnan Department of Electrical and Computer Engineering Ryerson University.

[13] A Study and Implementation of the Huffman Algorithm Based on Condensed Huffman Table, by Bao Ergude, Li Weisheng, Fan Dongrui, Ma Xiaoyu, School of Software, Beijing Jiaotong University; and Key Laboratory of Computer System and Architecture, Institute of Computer Technology, Chinese Academy of Sciences, 2008.

[14] An efficient Memory Construction Scheme for an Arbitrary Side Growing Huffman table, by Sung-Wen Wang, Shang-Chih Chuang, Chih-Chieh Hsiao, Yi-Shin Tung and Ja-ling Wu CMLab, Dept. CSIE NTU, Setabox Corporation, Graduate Institute of Networking and Multimedia NTU, NOVEMBER 2008.

[15] A Novel VLSI Architecture for Image Compression Model using Low Power DCT. By Vijaya Prakash A M and K. S Gurumurthy, WASET Vol .72 December 2010 Singapur.

[16] Balance of 0, 1 Bits for Huffman and Reversible Variable-Length Coding, by Jia-Yu Lin, Ying Liu, and Ke-Chu Yi, MARCH 2004.

[17] *Parallel Zigzag Scanning and Huffman Coding for a GPU-Based MPEG-2 Encoder*, by Pablo Montero, Javier Taibo Videalab University of A Coruna, A Coruna, Spain. Victor Gulias, MADS Group University of A Coruna, A Coruna, Spain. Samuel Rivas LambdaStream S.L. A Coruna, Spain.

[18] Discrete Wavelet Transform for Image Compression and A Model of Parallel Image Compression Scheme for Formal Verification…Proceedings of the World Congress on Engineering 2007 Vol I by Kamrul Hasan Talukder and Koichi Harada.

[19] Adaptive Context Based Coding for Losless Color Image Compression , by Yongli Zhu and Zhengya XU School of Computer Science and Technology North China Electrical Power University. IMACS Conference on 'Computational Engineering in Systems Applications "(CESA) 2006 Beijing China.

[20] An Efficient and Selective Image Compression Scheme using Huffman and adaptive Interpolation by Sunil Bhushan and Shipra Sharma.24th International Conference Image and Vision Computing New Zealand (IVCNZ 2009).

[21] Low Bit Rate Image Coding Based on Wavelet Transform and Color Correlative Coding by Wenna Li and Zhaohua Cui "2010 International Conference on Computer design and Applications (ICCDA 2010).

[22] Medical Image Coding based on Wavelet transform and Distributed arithmetic Coding by Li Wenna ,

Gao Yang , Yi yufeng and Gao Liqun. IEEE 2011 Chines Control and Decision Conference (CCDC).

Authors

Vijaya Prakash A.M Obtained is B.E from UVCE Bangalore of Bangalore University in the year 1992, Post graduated in M.E from SDMCET Dhawad of Karnataka University in the year 1997 and presently pursuing Ph.D. from Dr. M. G. R University Chennai. He has been actively guiding PG and UG student's projects in the area of VLSI Design and Image Processing. He has around 8 technical paper publications in international journals and international conferences. Currently he has been working as Associate Professor in Electronics and Communication Engineering Department, Bangalore Institute of Technology Bangalore-04. He has presented paper in Singapur. His research interests are Low Power VLSI, Image Processing, Retiming and Verification of VLSI Design, Synthesis and Optimization of Digital Circuits. He is a member of IMAPS and ISTE.

K.S Gurumurthy obtained his B.E degree from M.C.E – Hassan of Mysore University in the year 1973. He got his M.E Degree from University of Roorkee (now IIT-Roorkee) in 1982. He joined UVCE in 1982 and he has since been teaching Electronics related subjects. He obtained his Ph.D. degree in 1990 from IISc Bangalore. Presently he is a professor in the DOS in E & CE, UVCE, BU, Bangalore-1. He is a "University gold medal" winner from University of Roorkee and a recipient of the "Khosla award" for the best technical paper published in 1982. He has successfully guided 4 Ph.D., 2 M.Sc-Engg (by research) candidates and guided a number of UG and PG projects. He has around 75 technical paper publications in journals and international conferences. He has presented papers in JAPAN, FRANCE, MALAYASIA and USA.His interests are Low power VLSI, Multi valued logic circuits, Deep submicron Devices. He is a member of IEEE and ISTE.