ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

A COST AND POWER EFFICIENT MEDICAL IMAGE COMPRESSOR VLSI DESIGN WITH FUZZY DECISION AND BLOCK PARTITION FOR WIRELESS SENSOR NETWORKS

Ch. Venkateshwarlu [1], Kondapuram Ramyasri[2], Konyala Avanthi [2], Kunta Ruchitha[2], Macharla Jaya[2]

[1,2]Department of Electronics and Communication Engineering

[1,2]Malla Reddy Engineering College for Women (A), Maisammaguda, Medchal, Telangana.

ABSTRACT

This paper presents a novel hardware-oriented image compression algorithm and its very large-scale integration (VLSI) implementation for wireless sensor networks. The proposed novel image compression algorithm consists of a fuzzy decision, block partition, digital halftoning, and block truncation coding (BTC) techniques. A novel variable-size block partition technique was used in the proposed algorithm to improve image quality and compression performance. In addition, eight different types of blocks were encoded by Huffman coding according to probability to increase the compression ratio further. In order to achieve the low-cost and low-power characteristics, a novel iteration-based BTC training module was created to get representative levels and meet the requirement of wireless sensor networks. A prediction and a modified Golomb-Rice coding modules were designed to encode the information of representative levels to achieve higher compression performance.

Keywords: Image compression, very large-scale integration, fuzzy decision.

1. INTRODUCTION

1.1 Compressor Algorithm

The Compressor algorithm was introduced in 1967 as an efficient method for decoding convolutional codes [1], widely used in communication systems [2].This algorithm is utilized for decoding the codes used in various applications including satellite communication, cellular, and radio relay. It has proven to be an effective solution for a lot of problems related to digital estimation. Moreover, the Compressor decoder has practical use in implementations of high-speed (5 to 10 Gb/s) serializer-deserializers (SERDESs) which have critical latency con-straints. SERDESs can be further used in local area and synchronous optical networks of 10 Gb/s. Furthermore, they are used in magnetic or optical storage systems such as hard disk drive or digital video disk [3].

The Compressor algorithm process is similar to finding the most-likely sequence of states, resulting in sequence of observed events and, thus, boasts of high efficiency as it consists of finite number of possible states [4–7]. It is an effective implementation of a discrete-time finite state Markov process perceived in memory less noise and optimality can be achieved by following the maximum-likelihood criteria [8]. It helps in tracking the stochastic process state using an optimum recursive method which helps in the analysis and implementation [9, 10].

2. LITERATURE SURVEY

A top-level architecture for Compressor decoders is shown in Fig. 1.1. As seen in this figure, Compressor decoders are composed of three major components: branch metric unit (BMU), add-compare-select (ACS) unit, and survivor path memory unit (SMU). BMU generates the metrics corre-sponding to the binary trellis depending on the received signal, which is given as input to ACS which, then, updates the path metrics. The survival path is updated for all the states and is stored in the

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

additional memory. SMU is responsible for managing the survival paths and giv-ing out the decoded data as output.

BMU and SMU units happen to be purely forward logic. ACS recursion consists of feedback loops; hence, its speed is limited by the iteration bound [11]. Hence, the ACS unit becomes the speed bottleneck for the system. M-step look-ahead technique can be used to break the iteration bound of the Compressor decoder of constraint length K [12–18]. A look-ahead technique can combine several trellis steps into one trellis step, and if M > K, then throughput can be increased by pipelining the ACS architecture, which helps in solving the problem of iteration bound, and is frequently used in high-speed communication systems.

3. PROPOSED METHOD

Introduction

Recently, most data are produced by human activities such as typing, recording, picturing, scanning, etc. Because of the limitation of attention, accuracy, and time for the human, it is impossible to collect multiple data all the time. A solution to solve the problem is using sensors to substitute the human and collect data around human life [1, 2]. With the analysis of data collected by various sensors, the computer or server can trace or quantize some activities rather than just making decisions according to only one kind of information. With rapid development of sensors and wireless communication techniques, wireless sensor networks (WSNs) [3, 4] get people's attention. A WSN is a network formed by a large number of sensor nodes, and each node has one or multi sensors to detect physical phenomena [5] such as light, heat, pressure, etc. Since the amount of image data is much more than other data in the WSNs, the amount of image and video data grows as the number of wireless camera nodes [6, 7] in the WSNs. One of the most crucial problem is how to store and transmit images by using the limited storage and wireless bandwidth without dropping the image quality too much. Image compression is an efficient way to reduce the amount of image data for storage and transmission. The most popular still image compression method is joint photographic experts group (JPEG) standard [8]. The methods used in JPEG are converting each image from the spatial domain to the frequency domain with a mathematical operation called discrete cosine transform (DCT). Since human visual system is more sensitive to the change of low-frequency information, JPEG discards high-frequency information to make data smaller. Finally, the data is encoded by using a Huffman coding algorithm to increase the compression ratios. Since the JPEG compresses images by using DCT which requires high computational complexity, the hardware cost and power consumption of the JPEG encoder design is unsuitable for the WSNs. JPEG 2000 [9] was also developed by the joint photographic experts group with the intention of replacing their original DCT-based JPEG standard with wavelet transform. In addition, JPEG 2000 used a more sophisticated entropy encoding scheme to gain compression ratio over JPEG. The JPEG 2000 based methods are unsuitable for hardware implementation due to the high complexity of the wavelet transform. JPEG-LS [10] used a predictive scheme based on the three nearest neighbors and an entropy coding to compress still images. The characteristic of JPEG-LS is transform-free based image compression technique and JPEG-LS is popular used in medical image compression applications due to the characteristic of lossless. Without mathematical operation of transformation, the speed of compression of JPEG-LS is much faster and the computational complexity is also lower than JPEG and JPEG 2000. Nevertheless, the compression ratios of JPEG-LS are much less than those of JPEG and JEPG 2000. Hence, it is not suitable to be applied in the WSNs due to the limitation of wireless bandwidth. After analysis of JPEG, JPEG 2000, and JPEG-LS, it is difficult to find a suitable candidate which has both benefits of low complexity and high compression ratios for the WSNs. Delta modulator [11] and delta sigma modulator [12] were used in CMOS image sensors to compress images by selecting the pixel values to the inputs of delta modulators or delta sigma modulators. A single-shot

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

compressed sensing architecture for CMOS image sensor was developed in [13], in which the captured pixel values of the CMOS image were selected randomly. A block-based compressive sensing CMOS image sensor [14] was implemented, which had the benefit of costefficient. An adaptive non-uniform sampling delta modulation (ANS-DM) technique [15] was proposed to apply in audio and image processing. The delta modulation and sigma modulation based compression methods provided an efficient way to reduce the output data of the CMOS image sensor to achieve image compression. Block truncation coding (BTC) based method [16] is recognized as a potential candidate. Recently, there are many techniques proposed to improve the image quality and the compression ratios of BTC such as, source encoding of the outputs of a block truncation coder (BTC) with Vector Quantization (VQ) [17]; encoding the bitmap with a LUT-based VQ encoder and getting representation levels based on direct binary search [18]; and reducing coding rates with Hamming codes and a differential pulse code modulation (DPCM) [19]. A human visual system (HVS) based on a digital halftoning technique [20] was used to increase the efficiency of the BTCbased algorithm, which had benefits of low-complexity and high performance. Although, these methods advanced the performance of BTC method, they also lost the benefit of low complexity in the BTC. Hence, an ultra-cost image compression design based on BTC was developed in [21]. Although the hardware cost and performance are suitable for the WSNs, the fixed size of block limited the compression ratios. Hence, it is necessary to develop a more flexible, higher compression ratio, and lower complexity image compressor design for the WSNs

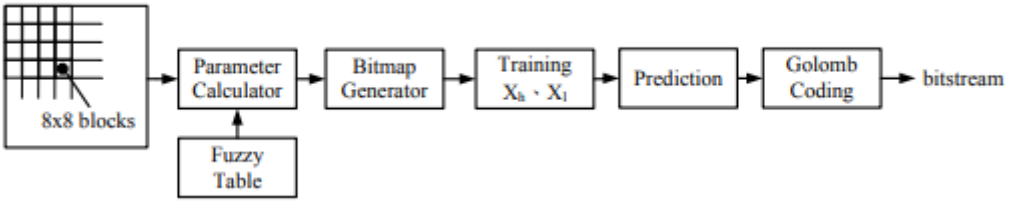


Fig. 2. Flowchart of the proposed variable-size block partition with fuzzy decision BCT-based image compression algorithm.

THE PROPOSED IMAGE COMPRESSION ALGORITHM

Block truncation coding (BTC) [16] was proposed by Delp and Mitchell in 1979. In BTC algorithm, each image is divided into non-overlapping M×N sub-images. Each sub-image is expressed by one bitmap and two representative levels. Unlike other modern compression techniques such as JPEG [8] based on discrete cosine transformation and JPEG-2000 [9] based on wavelet transformation, BTC is a transform-free still image compression algorithm. It is one feature in BTC-based algorithm. Because of that, it also has other features such as low-complexity, low-memory-requirement and easyimplementation for hardware realization. Nevertheless, the traditional BTC algorithm has one significant problem, which has low compression ratios. For this reason, this study proposes a variable size block, fuzzy decision, and iteration-based BTC techniques to increase the compression ratios and the quality of the reconstructed image. Fig.1 shows the flowchart of the proposed variable-size block partition with fuzzy decision BCT-based image compression algorithm. Details of each partition were illustrated as follows:

A. Fuzzy Table

In previous BTC-based image compression algorithm [21], the compression ratios are not good enough for fixed-size 4×4 BTC compared with JPEG. Although compressing the image with fixed-size 8×8 BTC made compression ratios become better than JPEG, it caused the quality of the reconstructed images worse than fixed-size 4×4 BTC and JPEG. For smooth area, it is possible to choose bigger block to get better result on compression ratios. For non-smooth area, choosing bigger block loses more

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

information from original image. In that case, selecting smaller block will be a wise choice for the consideration of image quality. Hence, it is efficient to find suitable block sizes for different situations, which can find the best block size for BTC-based image compression algorithm. In order to simplify the block-type selection in the encoding process, eight block-types were used in the proposed algorithm as shown in Fig. 2. The block-type can be obtained by looking up fuzzy table with parameter f and parameter g described below. Fig. 3 describes the difference between image, sub- image, and sub-block. The sub-image in Fig. 3 was assumed a block-type G partition. Thus, this sub-image has three subblocks; two 4×4 blocks and one 8×4 block. Each sub-block was sent to training module to get the representative levels

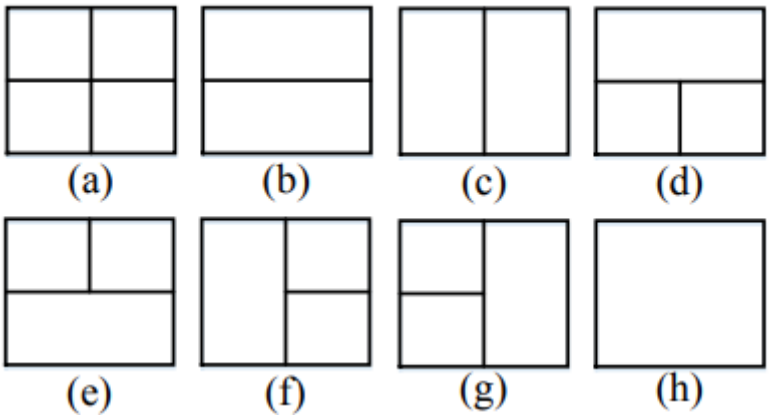


Fig. 3: Eight block-types of the proposed BCT-based image compression algorithm. (a) Block-type A (b) Block-type B (c) Block-type C (d) Block-type D (e) Block-type E (f) Block-type F (g) Block-type G (h) Block-type H

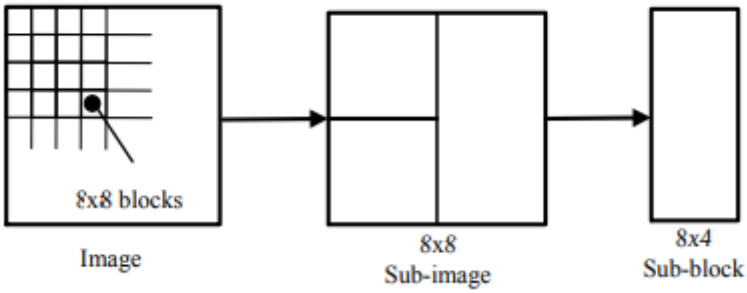


Fig. 4: Relationship between image, sub-image, and sub-block.

To acquire a win-win result, a fuzzy theory was used for encoder to select block-type for representative levels training and bitmap production. In order to find the parameters of fuzzy theory, eight images of the Kodak dataset were used to train the parameters of the fuzzy table. Fig. 4 shows the eight training images for the fuzzy table and Table I lists the trained parameters of the fuzzy table. The processing steps of the fuzzy table and the parameters of fuzzy logic control are discussed below.

Step 1: Calculating means of each 4×4 sub-block in the same 8×8 sub-image.

Step 2: Summing up these mean values, then divided by 8. Taking that as parameter f (in Table I).

Step 3: Calculating variance of each 4×4 sub-block. Taking that as parameter g (in Table I).

Step 4: Recording the block-type which has the lowest MSE and the best compression ratio, also recording parameter f and parameter g.

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

Step 5: In the end, one of eight block-types has win-win result will be found out as a specific parameter interval.

After the process described above, Table I was ready for encoder to find suitable block sizes for different situations. For example, if f is Medium and g is Very High, the partition of subimage is selected as block-type D. Table I maps the two input fuzzy sets to an output fuzzy set. Therefore, Table I contains fuzzy rules, and is also called fuzzy table in this paper.

TABLE I
FUZZY TABLE TRAINED BY THE EIGHT TESTING IMAGES

f \ g	Ultra-Low	Very Low	Low	Medium	High	Very High	Ultra-High
Ultra-low	H	H	H	H	H	C	A
Very Low	H	B	D	H	C	B	A
Low	H	B	A	C	H	A	H
Medium	H	B	E	H	A	D	A
High	H	C	B	H	H	B	A
Very High	H	C	E	H	F	B	H
Ultra-high	H	H	B	H	B	F	G

TABLE II
THE ORIGINAL BINARY CODES, HUFFMAN CODES, AND PROBABILITY FOR EIGHT BLOCK-TYPES IN THE FUZZY TABLE

Block-type	Original binary codes	Huffman Codes	Probability
A	000	001	14.29%
B	001	000	18.37%
C	010	011	10.20%
D	011	01010	4.08%
E	100	01001	4.08%
F	101	01000	4.08%
G	110	01011	2.04%
H	111	1	42.86%
Average length	3	2.43	

Table I shows an example of eight block-types in the fuzzy table. It is easy to find that the probability of eight block-types are much different. Hence, to further improve the compression ratio, the binary codes represent eight block-types should be encoded by Huffman coding. Table II lists the original binary codes, Huffman codes, and probability for eight block-types. The average length of original binary and Huffman coding are 3-bits and 2.43-bits, respectively, which means more than 19% bits are reduced by the Huffman coding for eight block-types

B. Bitmap

The bitmap is an image after binarization. Hence, there are only two representative levels to reconstruct the decoding images. The representative levels show the intensities of all block pixels in the decoded image. Thus, it is important to find optimized representative levels for each block. To get the bitmap of each sub-block, the average value of all pixels in a sub-block is used as a threshold to binarize values in sub-block, as shown in Eq. (1). $Y(i,j) = \{1, if\ X(i,j) \geq mean\ 0, otherwise$ (1) where Y(i, j) is the binarized value of X(i, j). In the decode process, the decoded pixel K(i, j) can be calculated by $K(i,j) = \{\ xh, if\ I(i,j) = 1\ xl\ , otherwise$ (2) where xh is the high-level representative value and xl is the low-level representative value in the corresponding sub-block and they can be obtained from training. HVS is more sensitive to the change of brightness. Green color also has more proportion of brightness than red and blue colors. For this reason, the bitmaps of the RGB images are quite similar as result of high correlations between each other. Therefore, only green color of bitmaps is in use for both encoding and decoding, which decreased more than 40% total bit rates.

C. BTC Parameters

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

Training In traditional BTC, xh and xl can be obtained from mean and standard deviation. However, the calculation of standard deviation involves square root and multiplication, that is unsuitable for VLSI implementation. Hence, a modified lowcomplexity representative levels which were used to substitute the bitmap in decoder can be obtained by $xh = max - \alpha 1(max - mean)$ (3) $xl = min + \alpha 2\ (mean - min)$ (4) where max is the maximum pixel value and min is the minimum pixel value in the corresponding sub-block. Details of the representative levels training steps are illustrated as follows: Step 1: Setting min_MSE as max value. Evaluates compressed image with Mean Square Error(MSE). $MSE = \sum[X(i,j) - K(i,j)]$ 2 (5) where X(i,j) and K(i,j) are the original value and the reconstructed value of the pixel in the same position. Step 2: Varying the value of $\alpha 2$ from 0 to 1. The values of $\alpha 2$ are increased by 0.25 in each cycle. The value of MSE is calculated and compared with min_MSE in each cycle. If MSE is smaller than min_MSE, min_MSE is replaced by MSE and record its corresponding low level xl . Step 3: After complete the training of the low level xl , a new training procedure is start to vary $\alpha 1$ from 0 to 1. The procedure to train xh is similar to the way how to train xl as described in step 1 and step 2.

D. Prediction

In traditional BTC, the compressed data consist of two 8-bit representative levels (xh and xl) and one binary bitmap corresponding to each sub-block. For smooth areas, a simple predictive coding method with entropy coding makes more effective to compression ratio. To reduce the complexity of prediction, every sub-blocks in the same sub-image share the same predictive values of xh and xl, which means that the two predictive values of the representative levels can be reshaped into a 64×64 matrixes in each color map. The predictive value was assigned by predictor with neighboring matrices as shown in Fig. 5. Eq. (6) shows the method to calculate the predicted value of each representative level xmed.

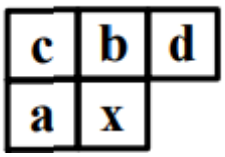


Fig. 5: Neighboring matrices for prediction.

$xmed$ = a+a+b+d 4 (6)

Predictive method is totally independent to the training described previously. The pixel after ‘x’ cannot be obtained until the pixel before ‘x’ was reconstructed. That is the reason why the calculation of predictor using the pixel after ‘x’ in decoding process will produce errors. In addition, the ‘d’ was selected because of using better average predictor [17] than JPEG-LS predictor.

E. Modified GR Coding

Since most of prediction errors concentrate between ±50, encoding prediction errors with GR coding could make the image compression ratios increased further more.

a. Modified GR Encoding

The traditional GR coding algorithms are used to encode positive values only. A positive value is expressed as a prefix, segmentation and suffix codes with traditional GR coding.

Since prediction error is not always a positive value, a modified GR coding technique including a signed bit was used in [21]. It has also more effect on compression. In this method, the output code consisted of a prefix, segmentation, signed bit and suffix codes. An absolute of prediction error divided by 4 to produce the values of the quotient q and the remainder r. The prefix code is q bits of “0”. The

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

segmentation is always 1 bit of "1". The sign bit is "0" or "1" depending on the sign of prediction error. The suffix code is 2 bits of r in binary representation. The modified GR codes with a signed bit were listed in Table III. The modified GR coding has higher performance to be applied in image compression because the traditional GR coding takes more bits than signed GR coding to encode larger or smaller values. For example, when encoding 31 with the traditional GR coding and the modified GR coding, it takes 18- bits and 11-bits by traditional and signed GR coding, respectively.

TABLE III THE DIFFERENT BETWEEN TRADITIONAL GR CODES AND MODIFIED GR CODES

Number	Traditional GR Codes	Modified GR Codes
0	100	1000
-1	110	1101
1	111	1001
-2	0100	1110
2	0101	1010
	:	
-9	0000110	001001
9	0000111	001101
-10	00000100	001110
10	00000101	001010
	:	
-30	0000000000000100	00000001110
30	0000000000000101	00000001010
-31	0000000000000110	00000001111
31	0000000000000111	00000001011

b. Modified GR Decoding

Fig. 6 shows the flowchart of the decoding process of the modified GR. First, the number of bit "0" is counted before segmentation, the first bit "1", shows up. Second, the next 3 bits is identified as a sign bit and 2 bits of the remainder. Third, if absolute of prediction error is greater than or equal to 32, it won't take any advantage of entropy coding. Moreover, it takes more bits than the binary representation. This is the reason why the value greater than or equal to 32 was encoded with binary representation after an impossible circumstance code (negative zero encoded with signed GR coding) which is to tell decoder the following 8 bits are binary representation. Finally, if value is encoded in binary representation, it could be outputted directly. Otherwise, the sign information will be added to the final decoded value according to the decoded value of the sign bit.

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

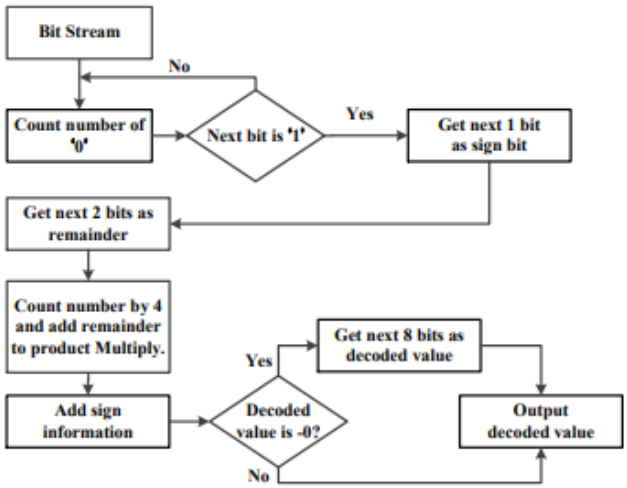
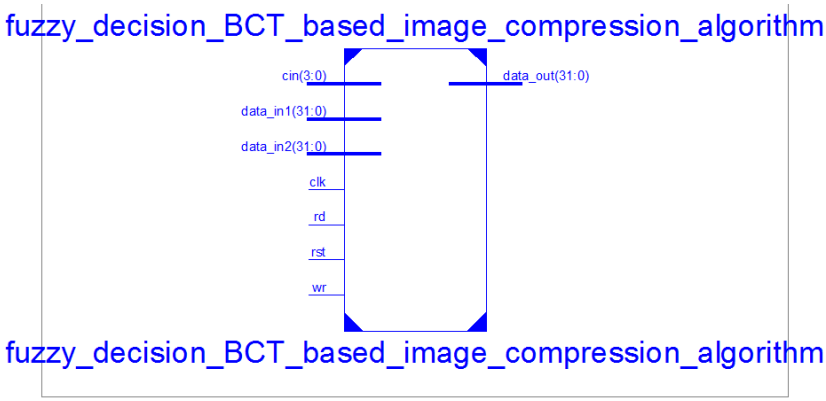


Fig. 6: Flowchart of the decoding process for the signed GR coding.

4. SIMULATION RESULTS



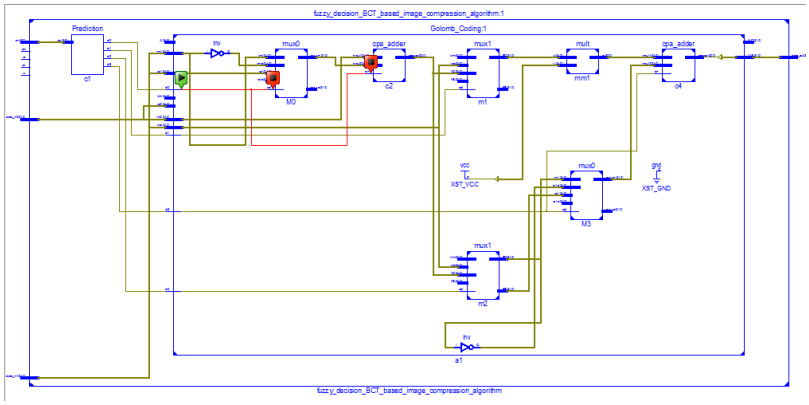


Fig. 7: RTL Schematic.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	1752	204000	0%
Number of fully used LUT-FF pairs	0	1752	0%
Number of bonded IOBs	100	600	16%

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

Fig. 8: Design summary.

```
LUT3:I0->O            1    0.043   0.289  a1/mm1/Madd_n06166 (a1/mm1/Madd_n06166)
LUT4:I3->O            1    0.043   0.000  a1/mm1/Madd_n0616_lut<0>25 (a1/mm1/Madd_n0616_lut<0>2
MUXCY:S->O            1    0.230   0.000  a1/mm1/Madd_n0616_cy<0>_24 (a1/mm1/Madd_n0616_cy<0>2
XORCY:CI->O           2    0.251   0.432  a1/mm1/Madd_n0616_xor<0>_25 (a1/mm1/n0616<26>)
LUT3:I0->O            1    0.043   0.289  a1/mm1/Madd_n06224 (a1/mm1/Madd_n06224)
LUT4:I3->O            1    0.043   0.000  a1/mm1/Madd_n0622_lut<0>27 (a1/mm1/Madd_n0622_lut<0>2
MUXCY:S->O            1    0.230   0.000  a1/mm1/Madd_n0622_cy<0>_26 (a1/mm1/Madd_n0622_cy<0>2
XORCY:CI->O           2    0.251   0.432  a1/mm1/Madd_n0622_xor<0>_27 (a1/mm1/n0622<28>)
LUT3:I0->O            1    0.043   0.289  a1/mm1/Madd_n06282 (a1/mm1/Madd_n06282)
LUT4:I3->O            1    0.043   0.000  a1/mm1/Madd_n0628_lut<0>29 (a1/mm1/Madd_n0628_lut<0>2
MUXCY:S->O            1    0.230   0.000  a1/mm1/Madd_n0628_cy<0>_28 (a1/mm1/Madd_n0628_cy<0>2
XORCY:CI->O           1    0.251   0.289  a1/mm1/Madd_n0628_xor<0>_29 (a1/mm1/Madd_prod_lut<30
LUT1:I0->O            1    0.043   0.000  a1/mm1/Madd_prod_cy<30>_rt (a1/mm1/Madd_prod_cy<30>_r
MUXCY:S->O            0    0.230   0.000  a1/mm1/Madd_prod_cy<30> (a1/mm1/Madd_prod_cy<30>)
XORCY:CI->O           1    0.251   0.542  a1/mm1/Madd_prod_xor<31> (a1/PCA<31>)
LUT5:I0->O            1    0.043   0.279  a1/c4/generate_N_bit_Adder[31].f/Mxor_s_xo<0>1 (data_
OBUF:I->O                  0.000          data_out_31_OBUF (data_out<31>)
----------------------------------------
Total                     14.327ns (4.923ns logic, 9.404ns route)
                                   (34.4% logic, 65.6% route)
```

Fig. 9: Time summary.

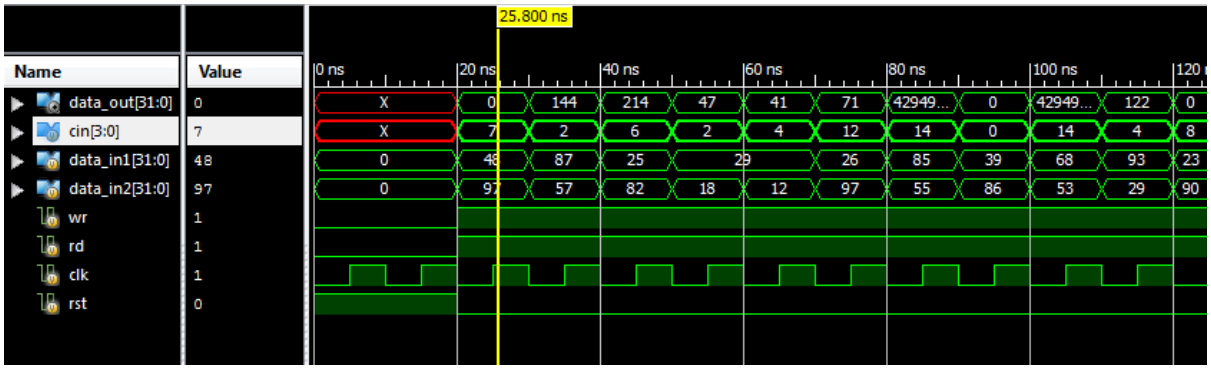


Fig. 10: Simulation outcome.

Device		On-Chip	Power (W)	Used	Available	Utilization (%)
Family	Virtex6	Clocks	0.000	1	---	---
Part	xc6vlx75tl	Logic	0.000	46	46560	0
Package	ff484	Signals	0.000	122	---	---
Temp Grade	Commercial	IOs	0.000	52	240	22
Process	Typical	Leakage	1.065			
Speed Grade	-1L	Total	1.065			

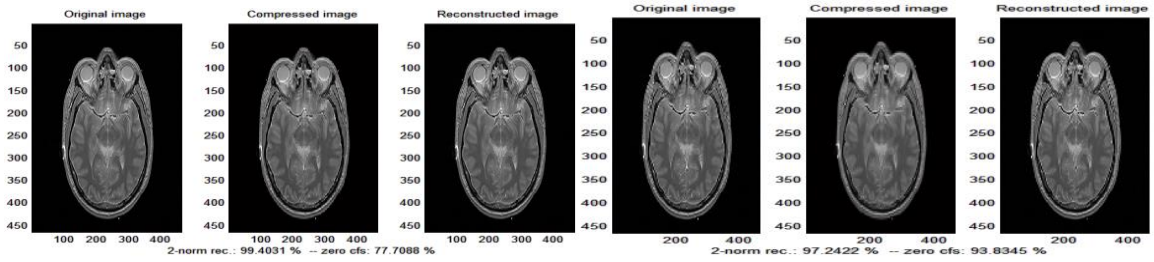
Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	0.900	0.435	0.000	0.435
Vccaux	2.500	0.045	0.000	0.045
Vcco25	2.500	0.001	0.000	0.001
MGTAVcc	1.000	0.303	0.000	0.303
MGTAVtt	1.200	0.213	0.000	0.213

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.7	82.1	52.9

Supply Power (W)	Total	Dynamic	Quiescent
	1.065	0.000	1.065

Fig. 11: Power summary.



ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

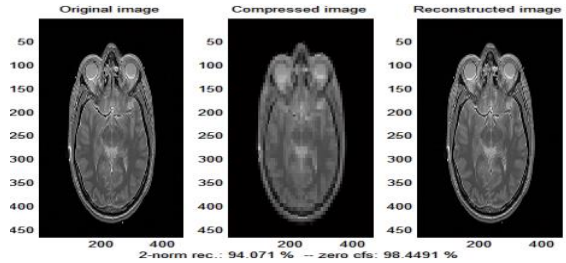


Fig. 12 Original, compressed and reconstructed images with *'haar'* 1st, 2nd and 3rd level of decomposition.

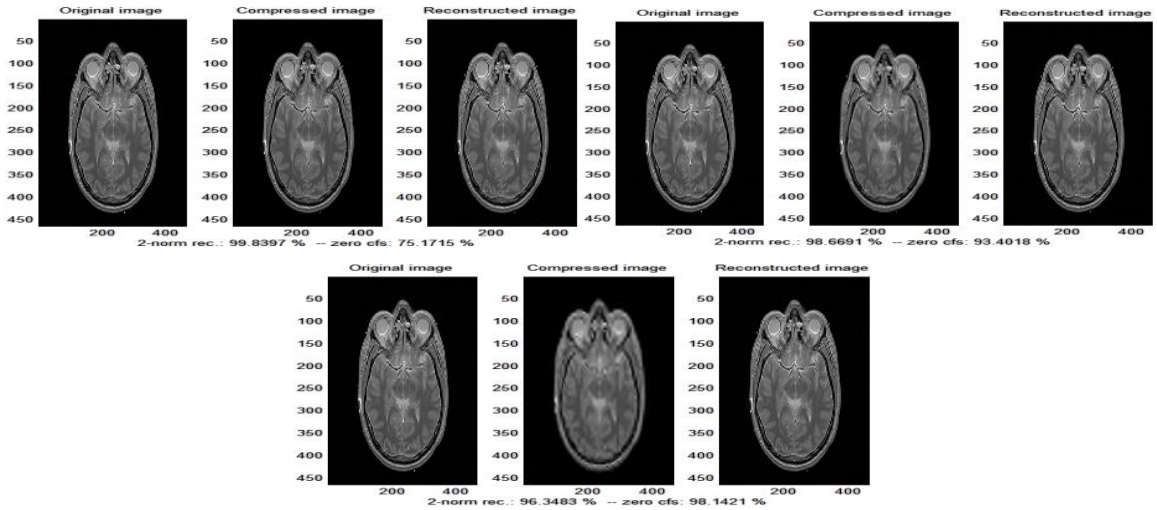


Fig. 13 Original, compressed and reconstructed images with '*bior4.4*' 1st, 2nd and 3rd level of decomposition.

5. CONCLUSION

In this work, a novel hardware-oriented, low-complexity, low-memory-requirement, and transform-free image compression algorithm consists of a digital halftoning, block truncation coding, fuzzy decision, and block partition techniques for VLSI implementation. A novel variable-size block and fuzzy decision techniques for block truncation coding was designed to improve the image quality and compression ratios. The block-type was encoded with Huffman coding to further increase compression ratios. An improved entropy encoder was developed to enhance the compression ratios. Compared with previous image compressor designs, this work reduced gate counts by at least 20.9% and had better FOM value than previous designs.

REFERENCES

[1]. Chen, Shih-Lun, et al. "VLSI implementation of a cost-efficient near-lossless CFA image compressor for wireless capsule endoscopy." *IEEE Access* 4 (2016): 10235-10245.

[2]. Asha, A., and S. M. Rajeshkumar. "Low Power Consumption using CMOS VLSI Design in Modern Trends."

[3]. Thilagavathi, K., and S. Sivanantham. "Two-stage low power test data compression for digital VLSI circuits." *Computers & Electrical Engineering* 71 (2018): 309-320.

[4]. Al-Ani, Muzhir Shaban. "Hardware Implementation of a Real Time Image Compression." *architecture* 28.29 (2017): 30.

[5]. Hsieh, Jui-Hung, Meng-Ju Shih, and Xin-Hao Huang. "Algorithm and VLSI architecture design of low-power SPIHT decoder for mHealth applications." *IEEE transactions on biomedical circuits and systems* 12.6 (2018): 1450-1457.

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

[6]. Rusyn, Bogdan P., Alexey A. Lutsyk, and Rostislav Ya Kosarevych. "Modified Architecture of Lossless Image Compression Based on FPGA for On-Board Devices with Linear CCD." *Journal of Automation and Information Sciences* 51.2 (2019).

[7]. Kasban, Hany, and Sherief Hashima. "Adaptive radiographic image compression technique using hierarchical vector quantization and Huffman encoding." *Journal of Ambient Intelligence and Humanized Computing* 10.7 (2019): 2855-2867.

[8]. Mrudula, S. T., KE Srinivasa Murthy, and MN Giri Prasad. "M-ABRC (Adaptive Binary Range Coder) using Virtual Sliding Window technique and its VLSI implementation." *Microprocessors and Microsystems* 71 (2019): 102901.

[9]. Das, Subhajit, Reshmi Maity, and N. P. Maity. "VLSI-based pipeline architecture for reversible image watermarking by difference expansion with high-level synthesis approach." Circuits, Systems, and Signal Processing 37.4 (2018): 1575-1593.

[10]. M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEGLS," IEEE Transactions on Image Processing, vol. 9, no. 8, pp. 1309– 1324, Aug. 2000.

[11]. Krishnaswamy, R. "VLSI based orthogonal diagonal cross hair search (ODCHS) algorithm implementation for efficient image compression." *Microprocessors and Microsystems* (2020): 103114.

[12]. Nagaraj, P., et al. "VLSI Implementation of Image Compression using TSA Optimized Discrete Wavelet Transform Techniques." *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020.

[13]. Tadisetty, Srinivasulu. "A novel ortho normalized multi-stage discrete fast Stockwell transform based memory-aware high-speed VLSI implementation for image compression." *Multimedia Tools and Applications* 78.13 (2019): 17673-17699.

[14]. Mukkara, Lakshmi Kiran, and K. Venkata Ramanaiah. "A Simple Novel Floating Point Matrix Multiplier VLSI Architecture for Digital Image Compression Applications." *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE, 2018.

[15]. Lahane, Aparna, and V. B. Malode. "VLSI Implementation of Daubechies Wavelet Filter for Image Compression."

[16]. kiran Mukkara, Lakshmi, and K. Venkata Ramanaiah. "CSD Based VLSI Architecture for NN Based Image Compression."

[17]. Muthukumaran, N., and R. Ravi. "Hardware implementation of architecture techniques for fast efficient lossless image compression system." *Wireless Personal Communications* 90.3 (2016): 1291-1315.

[18]. Soorya, B., S. Sweetline Shamini, and K. Sangeetha. "VLSI Implementation of Lossless Video Compression Technique Using New Cross Diamond Search Algorithm." *International Journal of Communication and Computer Technologies, 5 (1), 27* 31 (2017).

[19]. Premachand, D. R., and U. Eranna. "Sector-Selective Hybrid Scheme Facilitating Hardware Supportability Over Image Compression." *Computer Science On-line Conference*. Springer, Cham, 2020.

[20]. Safieh, Malek, and Jürgen Freudenberger. "Efficient VLSI architecture for the parallel dictionary LZW data compression algorithm." *IET Circuits, Devices & Systems* 13.5 (2019): 576-583.

[21]. S.-L. Chen, J. Nie, T.-L. Lin, R.-L. Chung, C.-H. Hsia, T.-Y. Liu, S.-Y. Lin and H.-X. Wu, "VLSI implementation of an ultra-low-cost and lowpower image compressor for wireless camera networks," Journal of RealTime Image Processing, Dec. 2015.

ISSN: 0975-3583,0976-2833 VOL14, ISSUE 07, 2023

[22]. Chen, Shih-Lun, and Guei-Shian Wu. "A cost and power efficient image compressor VLSI design with fuzzy decision and block partition for wireless sensor networks." *IEEE Sensors Journal* 17.15 (2017): 4999-5007.

[23]. Coutinho, Vítor A., et al. "Pruned discrete Tchebichef transform approximation for image compression." *Circuits, Systems, and Signal Processing* 37.10 (2018): 4363-4383.

[24]. Turcza, Paweł, and Mariusz Duplaga. "Energy-efficient image compression algorithm for high-frame rate multi-view wireless capsule endoscopy." *Journal of Real-Time Image Processing* 16.5 (2019): 1425-1437.

[25]. Rusyn, Bogdan, et al. "Lossless image compression in the remote sensing applications." *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)*. IEEE, 2016.

[26]. Chetan, H., and G. Indumathi. "Low Power VLSI Implementation of Data Compression for Multimedia Devices using CDF m/n DWT on to Resource Constrained Dynamically Reconfigurable Memories."

[27]. C.-J. Lian, L.-G. Chen, H.-C. Chang, and Y.-C. Chang, "Design and implementation of JPEG encoder IP core," Proceedings of the ASP-DAC 2001. Asia and South Pacific Design Automation Conference 2001 (Cat. No.01EX455), 2001.

[28]. Z. Qihui, C. Jianghua, Z. Shaohui, and M. Nan, "A VLSI implementation of pipelined JPEG encoder for grayscale images," 2009 International Symposium on Signals, Circuits and Systems, Jul. 2009.

[29]. P. Merlino and A. Abramo, "A fully pipelined architecture for the LOCOI compression algorithm," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 17, no. 7, pp. 967–971, Jul. 2009.

[30]. X. Xie, G. Li, X. Chen, X. Li, and Z. Wang, "A low-power digital IC design inside the wireless endoscopic capsule," IEEE Journal of SolidState Circuits, vol. 41, no. 11, pp. 2390–2400, Nov. 2006.