# Hardware Architecture for Fractal Image Encoder with Quadtree Partitioning

Mamata Panigrahy
Department of E&ECE
IIT Kharagpur, India

Indrajit Chakrabarti
Department of E&ECE
IIT Kharagpur, India

Anindya Sundar Dhar
Department of E&ECE
IIT Kharagpur, India

## ABSTRACT
This paper presents the hardware architecture for fractal image compression (FIC) with quadtree partitioning. Fractal image coding with quadtree partitioning allows one to produce higher quality of image. Processing image areas of different complexity with image blocks of varying size enables proper exploration of image details. Additionally, exploiting parallelism present within the algorithm and adopting hardware based solutions speed up the encoding process. The proposed architecture has been implemented on Xilinx Vertex-5 FPGA operating at a frequency of 154MHz.

## General Terms
Image compression, Fractal encoder

## Keywords
FIC; PIFS; PSNR; Morton scan.

## 1. INTRODUCTION
Image compression is one of the basic needs in the growing multimedia and web applications. Image archival and its assessment require high quality of image with high speed of downloading. Among several image compression methods, fractal image compression is one of the promising method that inherits the property of high compression with good image quality at low bit rates along with fast decompression speed. Again, image can be viewed at varying resolution and its processing without requiring codebook as in vector quantization makes the scheme more attractive for its applications. Generating complex images with self-similarity at each level is best fit for this compression method. This technique finds its application in the areas of image watermarking, feature extraction, medical applications and many more. However, the main obstacle of this technique for its real time applications is its long encoding procedure. The encoding procedure deals with nearly all sorts of operations followed in image processing that includes image segmentation, affine transformation, thresholding, searching and distortion measuring mechanism that needs a long time to complete the whole process.

Jacquin [1] proposed the full automated algorithm for fractal image encoder. Fractal image compression process is based on the theory of partitioned iterated function system (PIFS). Instead of finding similarity in the whole image, this technique segments the image into small sized blocks and tries to find the similarity between these sub-blocks. Similar areas are marked with their positions and parameters of the best matching blocks are stored without any direct image information. This needs quite a small space for representing the whole image and leads to higher compression. All the similar blocks are related by affine transformations that are contractive in nature. Regeneration of the original image is performed by applying these transformation parameters iteratively on any arbitrary image. This enables one to generate the original image from the virtual codebook parameters without prior knowledge of the original image. Therefore, while encoding, though different factors like compression ratio and the reconstructed quality requirement, plays important roles, the decoding step is independent of such factors and hence, the decoded image can be zoomed up to any scale. Hence, this lossy image compression scheme not only benefited with high compression but also converges to the original image quickly with a few iterations. Special advantage of this unsymmetrical encoding and decoding mechanism is seen while dealing with web applications and internet surfing.

The aim of this paper is to reduce the time complexity of the fractal image encoder. Researchers adopted both the algorithmic modifications and hardware based solutions to improve the encoder performance. Classification approach [1-2] reduces the overall processing time by searching the best matching domain block in the groups that contain domain blocks with similar features. These features may be based on the property of presence of edges, textures or considering the intensity of the block. Other researchers excluded the domain blocks that fails to satisfy certain statistical characteristics like mean and variance of the blocks. Nearest neighbor [3] and spatial correlation based approaches [4] also succeed to achieve speed up in the encoding process. Again, search-less schemes [5] are found in the literature that places the domain blocks to the neighboring area relative to the range block that have high probability of obtaining similarity. Some of the researchers adopted tree structure [6] to reduce the total encoding time in a logarithmic order of the image size $O(\log_2 N)$ for a $N \times N$ image. Various researches [2, 7] have been made on the basis of image partitioning techniques adopted to form the image blocks.

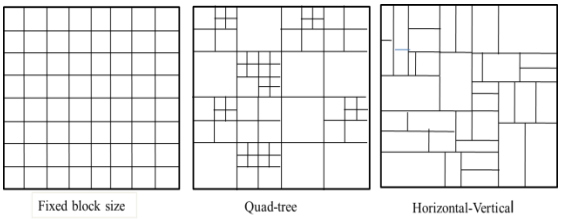


**Fig. 1: Regular partitioning schemes (a) Fixed size (b) Quad tree partitioning (c) HV partitioning.**

The block partitioning is made according to the intensity variation of the block or in a content dependent way as per the presence of smooth regions or edges. Accordingly, different size and shapes of the range blocks are found in the literature that may be regular (square, rectangular, triangular) or irregular shaped. Fig.1 shows different types of regular partitioning schemes.

Region based functionality [8] improves the compression performance while retaining the image quality. Further improvement in encoding speed can be achieved with hardware based solutions [9-14]. These processes exploit the parallelism present in the algorithm itself. Executing few independent operations in a concurrent manner saves the number of clock cycles while matching range and domain blocks. Similarly, operating in a pipelined manner improves the overall throughput of the system. Several efficient hardware designs are available in the literature that improved the overall time complexity of the FIC encoder. While encoding time reduction is the main aim of most of the designs, the hardware complexity of the encoder is another considering factor in most of the cases. Vidya proposed an integer based FIC design [9] that is based on fixed size partitioning. Jackson proposed a search-less architecture based on quadtree (QT) partitioning. Acken proposed a QT partitioning based architecture [10] that uses full search mechanism to locate the best matching domain block.

The proposed design based on quadtree partitioning and searches all the half-overlapped domain blocks present in any image. This method results in high quality image at an improved compression speed. Processing several operations in a concurrent manner enhances the encoding speed.

Organization of the paper is as follows. Section 2 provides the fractal coding procedure. The proposed hardware architecture design is presented in Section 3. The performance of the proposed design is evaluated in Section IV. Finally, the concluding remarks are given in Section V.

## 2. THEORETICAL BACKGROUND BEHIND FRACTAL IMAGE COMPRESSION ALGORITHM

Image segmentation is the first step of encoding of fractal image compression technique. This procedure divides the whole image into number of blocks of two resolutions named as the range and the domain blocks. Range blocks are smaller blocks without possessing any overlapping between them whereas the domain blocks are the larger blocks with certain steps of overlapping. The domain blocks are generally selected twice the range block size in both horizontal and vertical directions. Similarity between these areas is found when both the blocks are mapped with each other and both the blocks inherit similar intensity variations. The range and domain blocks contain $B\times B$ and $2B\times 2B$ pixels respectively. Hence, the range pool is constructed with $\left(\frac{N}{B}\right)^2$ number of equal-sized non overlapping range blocks. However, the number of domain blocks in the domain pool is dependent on the chosen overlapping pixels. A domain pool Ω with single pixel overlapping contains a maximum number of domain blocks. However, locating the best matching domain blocks by searching this large number of domain blocks in the domain pool incurs an unacceptable long time. This probes one to increase the overlapping area and usually, the same number of pixels as that of the range block size is used to trade-off between image quality and encoding time. Hence, a domain pool is formed with $\left(\frac{N}{B}-1\right)^2$ number of domain blocks. While searching for the best matching domain block, the number of searching operations increases to an order of$O(N^4)$. This is mainly responsible for the long FIC encoding time. Again, if the block size is chosen adaptively, this gives rise to image blocks of different sizes. This condition further degrades the compression performance as the number of range blocks with sizes less than that of level-0 increases.

The transformation for mapping the range and the domain block includes geometric contraction and the intensity variation. Geometric contraction of the domain block is performed to equalize the size of the same with the range block. Intensity variation is done through variation of the luminance and the contrast of the image block so as to match the intensity of the range block. This intensity variation is achieved through a scaling parameter *s* and luminance shifting is possible by addition of an offset value *o*. The main aim behind this is to reduce the intensity difference between the range and the transformed domain block.

These two parameters are responsible for gray level transform of the domain block and is given as

$$G(D) = sD + oI \qquad (1)$$

The scaling parameter is constrained within the range between zero and one and quantized before storing. Similarly, to reduce the biasing between the scaling and offset parameters, the mean value of the range block is used as the offset value that lies between 0 and 255. Thus the scaling and offset parameters require 2 bits and 8 bits respectively when stored in the fractal codebook. Additionally, to get better similarity, the pixel positions of the transformed domain block are varied in eight different ways. These variations are known as eight isometries that includes $90^o$, $180^o$ and $270^o$ rotations around the centre, horizontal, vertical, and the first and second diagonal reflections along with the actual orientation.

Different similarity measures like mean square difference (MSE), least square error (LSE), sum of absolute difference (SAD), mean absolute difference (MAD) are used to compute the difference between the range and the domain block. To get convergence at less number of iterations, the present work subtracts the block mean value from the corresponding blocks before performing the scaling operation. The distortion *E* between the range (*R*) and the intensity transformed domain block (*D*) is given by

$$E(R,D) = \sum_{i=1}^{N} \left\| \left(R - r_{avg}\right) - \left(s\left(D - d_{avg}\right)\right) \right\|^2 \qquad (2)$$

The lower the value of this difference, the better matching found between both the blocks.

The optimal scaling and offset values can be computed by differentiating expression (2) and expressed as

$$s = \frac{\langle R_i - r_{avg}\, I, D_i - d_{avg}\, I \rangle}{\left\| D_i - d_{avg}\, I \right\|^2} \qquad (3)$$

$$o = r_{avg} - s d r_{avg} \qquad (4)$$

Where $r_{avg}$ represent the mean intensity value of the range block $R_i$ and $d_{avg}$ for the mean value of domain block $D_i$. The operator $\langle \cdot \rangle$ is used to denote inner product.*I* representing a B×B matrix with all values equal to one; *s* and *o* are the optimal scaling and the offset parameters.

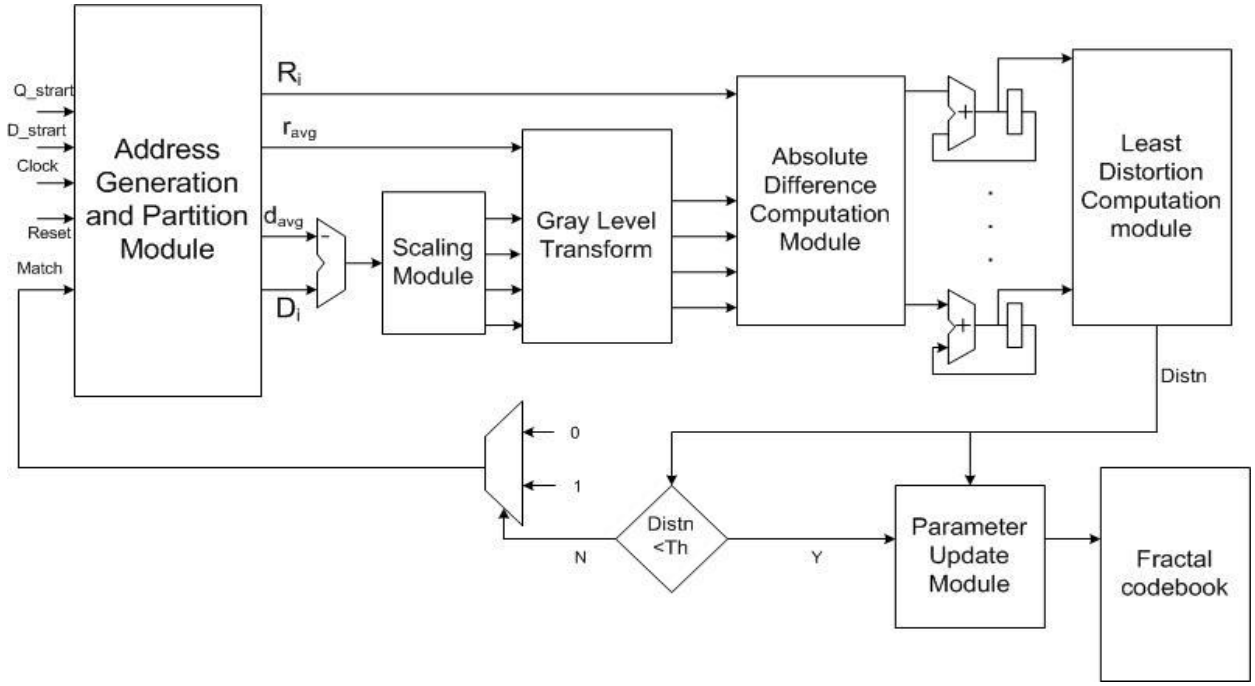


**Fig. 2: Block diagram of the proposed QT based fractal image encoder.**

## 3. PROPOSED ARCHITECTURE

The proposed architecture for the fractal image encoder uses the QT partitioning to obtain the range and the domain blocks. This structure uses three level of QT structure that presents range block sizes 8×8, 4×4 and 2×2 at level-0, level-1 and level-2, respectively. This necessitates domain blocks of sizes 16×16, 8×8, 4×4 in the respective levels. The input to the encoder is an image of size 256×256 pixels. Therefore, the range pool contains 1024 range blocks and 961 domain blocks at level-0. If the threshold criterion is not satisfied, the number of range and domain blocks is increased to 4096 and 3969 at level-1 and 16384 and 16129 at level-2, respectively.

The block diagram of the implementation is shown in the Fig. 2. The main modules of this design are the address generation and partition module, scaling module, the least distortion computing module, threshold matching unit, and parameter update module.

Dual port memory is used to supply the range and domain block pixels based on the address generated by the address generation module. The range and the domain blocks are fetched in a sequential manner while pixels inside the blocks are fetched in Morton scan order as shown in Fig. 3. For fetching a range block and a domain block at level-0, 16 and 64 clock cycles are required. One additional clock cycle is required to compute the mean value of the domain block. The range mean computation is performed concurrently during the domain pixel fetching operations. In the present work, two bits are allotted for the scaling parameter. The scaling module therefore generates four copies of the intensity transformed domain pixels with scaling factor of 0.25, 0.5, 0.75 and 1. Scaling by factors of 0.25, 0.5 can be achieved simply by right shift action of two bits, single bit of the transformed domain pixel. This does not require additional hardware. However, scaling by a factor of 0.75 requires an additional adder circuit that adds the 0.25 and 0.5 scaled outputs of the domain pixel. As the isometric transform unit consumes nearly 8 times the hardware requirement for a single domain block, this operation is skipped in this architecture.

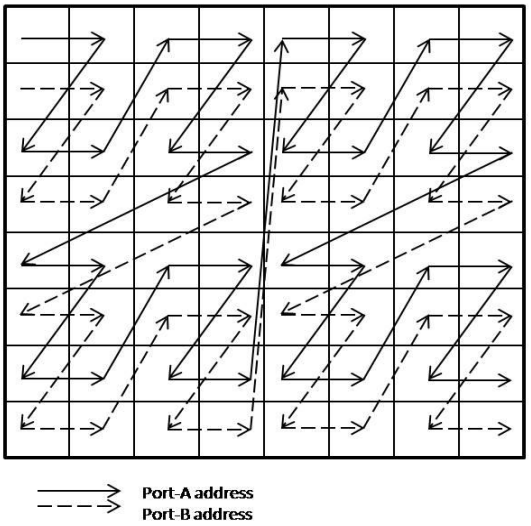


**Fig. 3: Morton scan order of pixel fetching.**

In order to avoid the multipliers for squaring operation in LSE measure, the proposed work uses MAD distortion measure to compute the differences. Seven additional clock cycles are required for computation of the SAD value for the four scaled domain blocks. Concurrent processing of the four scaled copies saves additional clock cycle consumption. Therefore, 73, 24 and 11 clock cycles are required at level-0, level-1, and level-2 respectively to complete the matching operation. If a close match is found, then further search operations are skipped and the fractal parameters are stored in the fractal codebook. Accordingly, the status of the flag Match indicates whether to opt for the next range block or search the next domain block. This signal is thereby given as an input to the address generation module to generate the address for the next fetching operation. The level signal is updated as all the domain blocks at any level have been searched and failed to satisfy the above criterion. A stop_partition signal is generated as all the range blocks of level-0 or level-1 are covered or all the range blocks at level-2 are attended. The

parameter for the least distorted domain block is stored in fractal codebook. The range block count is incremented to fetch the next range block.

# 4. RESULTS AND DISCUSSIONS

Fractal image coder with quadtree partitioning involves multi-resolution view of the image in the sense that smooth areas are covered with large block sizes and complex areas are covered with smaller block sizes. Due to recursive splitting of the range block till the user defined tolerance limit is met, the process involves long encoding time. As the threshold value is varied, the image quality, compression ratio as well as the time taken for encoding the image gets affected. The effect of variation of threshold value on the test image of Lena of size 256×256 is as shown in Table I.

**Table 1. Results for Lena image.**

| Th | PSNR(dB) | CR | Bpp | ET(s) | ET(h) |
|---|---|---|---|---|---|
| 1 | 43.30 | 1.95 | 4.10 | 7135 | 5.86 |
| 2 | 39.55 | 2.96 | 2.70 | 3848.6 | 3.64 |
| 4 | 35.18 | 4.85 | 1.64 | 1827 | 1.8 |
| 8 | 30.60 | 9.19 | 0.87 | 455.6 | 0.65 |
| 12 | 28.75 | 12.1 | 0.65 | 229.8 | 0.3 |

ET(s) - Software encoding time in sec.

ET(h) - Hardware encoding time in sec.

Bpp - bits per pixel.

PSNR- expressed in dB.

CR - Compression ratio.

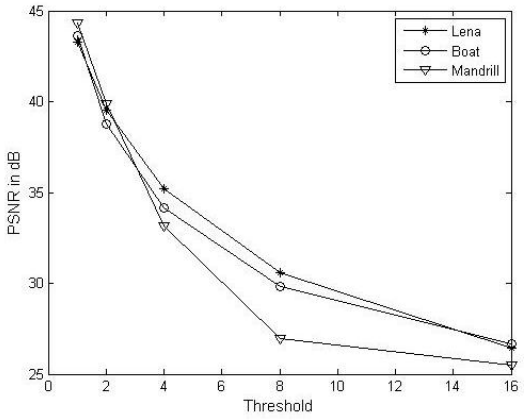


**Fig. 4: Effect of threshold on test images.**

Fig.4 depicts the effect of threshold on test images of different complexity. The experiment is performed on the medium complex Lena image and high complex Boat and Mandrill images. With increasing threshold value, the image quality degrades for all the test images. This is due to easy matching at lower QT levels. At higher threshold, the compression performance as well as the encoding speed improves at the expense of image quality. As the proposed design is based on the half domain overlapped full search approach, improved image quality is obtained compared to the search-less approaches proposed by Jackson [11] and Furao [5]. Again, the reconstructed image quality dominates over the classification approach proposed by Samavi [13]. However, due to large number of search operations, this approach cannot compete with the reduced encoding time due to the above mentioned approaches.

The proposed design has been coded with Verilog HDL, synthesized and simulated with Xilinx ISE and implemented on Xc5vlx50t-3ff1136 FPGA. The device utilization summary for the same is given in Table II. The proposed architecture can operate at a frequency of 154MHz.

**Table 2. Device utilization summary**

| Logic Utilization | Xc5vlx50t-3ff1136 |
|---|---|
| | % of Utilization |
| Number of slices registers | 7 |
| Number of slice LUTS | 22 |
| Slice LUT-FF pair | 15 |
| Number of block RAM/FIFO | 53 |

The design achieves an area advantage over the existing architectures by Jackson [11]. The proposed design outperforms the architecture due to Vidya [9] in terms of image quality and encoding time. Compared to the design [12-14] the proposed work achieves better image quality.

# 5. CONCLUSIONS

The paper presents the hardware implementation of quadtree based fractal image encoder that operates at a maximum clock frequency of 154 MHz. This architecture reduces the encoding time while limiting the hardware complexity. Again, the main advantage obtained from the proposed design is the image quality which is hardly achieved by the other techniques involving search-less methods and classification approaches. However, including speed enhancing approaches with the proposed design can improve the encoding time further to achieve the challenge of real time applications.

# 6. REFERENCES

[1] A.E. Jacquin, “Fractal image coding: a review”, Proc. IEEE Vol.81, no.10, Oct’1993, pp.1451–1465.

[2] Y. Fisher, Fractal Image Compression: Theory and Application, Springer, New York, (1994.

[3] C.Z. Tong and M. Wong, “Adaptive Approximate Nearest Neighbor Search for Fractal Image Compression”, IEEE Trans. Image processing, Vol. 11, no. 6, June 2002, pp. 605-615.

[4] T.K. Truong, C.M. Kung, J.H. Jeng and M.L. Hsieh, “Fast fractal image compression using spatial correlation”, Chaos, Solitons and Fractals 22,2004, pp. 1071–1076.

[5] S. Furao and O. Hasegawa, “A fast no search fractal image coding method”, Signal Processing: Image Communication, 19, 2004, pp. 393–404.

[6] B. Bani-Eqbal, “Speeding up fractal image compression”, Proc. SPIE: Still-Image Compression 2418, 1995, pp. 67–74.

[7] B. Wohlberg and G.D. Jager, “A review of the fractal image coding literature”, IEEE Trans. Image Process. Vol. 8, no.12, Dec 1999, pp.1716-1729.

[8] K.Belloulata and J.Konrad, “Fractal image compression with region based functionality”, IEEE Trans. Image processing, Vol. 11, no. 4, April 2002. pp. 351-362.

[9] D. Vidya, R. Parthasarathy, T.C. Bina and N.G. Swaroopa, “Architecture for fractal image compression”, J. Syst. Arch. 46, 2000, pp. 1275–1291.

[10] K P. Acken, M. J. Irwin and R. M. Owens, "A Parallel ASIC Architecture for Efficient Fractal Image Coding", Journal of VLSI Signal Processing 19, 1998, pp. 97–11

[11] D. Jackson, H. Ren, X. Wu and K.G. Ricks," A hardware architecture for real-time image compression using a search-less fractal image coding method". J Real-Time Image Proc. 1, 2007, pp. 225–237.

[12] M. Panigrahy, I. Chakrabarti, and A. Dhar, "VLSI design of fast fractal image encoder," in VLSI Design and Test, 18th International Symposium on, July 2014, pp. 1–2.

[13] S. Samavi, M. Habibi, S. Shirani, and N. Rowshanbin, "Real time fractal image coder based on characteristic vector matching," Image Vision Computing., vol. 28, no. 11, Nov. 2010, pp. 1557-1568.

[14] M. Panigrahy, I. Chakrabarti, and A. Dhar, "Low- delay parallel architecture for fractal image compression," Circuits, Systems, and Signal Processing, (CSSP), Vol. 35, no. 3, March 2016, pp.897-917.