Journal of Engineering and Applied Sciences 2 (3): 558-563, 2007

Architecture Proposed for New Method of Partitioning High Speed Fractal Image Compression

[1]B. Sankaragomathi, [2]L. Ganesan and [3]S. Arumugam
[1]National Engineering College, Kovilpatti-628 503, Tamil Nadu, India
[2]Department of CSE, Alagappa Chettiyar College of Engineering and Technology, Karaikudi, India
[3]AKCE, Srivilliputhur, Tamil Nadu, India

Abstract: The main problem of Fractal Image Coding (FIC) is the tremendous encoding time needed due to the large amount of comparisons between range and domain blocks. To overcome this problem, a few dedicated architectures proposed have utilized global data communication for providing domain blocks to all the processors. As the number of processors increases, expanding non-local communication paths is difficult without slowing down the system clock. In this study, we propose an efficient VLSI architecture for Fixed-size Partitioning Fractal Image Compression (FPFIC), which uses only local communication. The main feature of this architecture is that it is capable of performing the fractal image encoding without the external memory for the fixed domain blocks.

Key words: Partitioning fractal image compression, fast comparison module, Extended Right Domain Memory (ERDM), the Extended Bottom Domain Memory (EBDM) Extended Diagonal Domain Memory (EDDM)

INTRODUCTION

Fractal image compression techniques were originally proposed by Barnsley (1988) as a compression method for binary images and applied to gray-scale images by Jacquin (1992; 1993). Fractal coding has received considerable attention because of its use of the self-similarity in images, an innovation that had not been used previously in the field of image compression. Many improvements in compression efficient and decoded image quality have been reported (Davoine *et al*., 1995; Reusens, 1994; Zhao and Yuan, 1998).

The partitioning schemes are very important for fractal image compression (Reusens, 1994). There are many methods of partitioning an image based on squares and rectangles such as fixed-size partitioning, quad tree partitioning and HV partitioning (Fisher, 1995). Ancarani *et al*. (1996) designed an ASIC for fractal image compression that performs the eight comparisons between a range block and the transformed domain blocks in parallel utilizing data independence. To perform the image compression this architecture needs a host computer providing the chip inputs and storing the output. However, the area cost of this architecture is increased due to the increase of the circuit for parallel processing and the chip must provide all the domain blocks into the chip. A systolic array (Meal and Conway, 1980; Foster and Kung, 1980) is capable of computing the multiplication of pixels of range and domain blocks in parallel. This architecture has the external memory in order to store the rotating and/or flipping all the domain blocks. Fatemi and Panchanathan (1997) proposed a parallel and pipelined architecture called “Fractal Engine". Affine module in Fractal Engine is capable of executing the eight isometric transformations on a domain block by Reflection Unit and Transposer Unit. All architectures in literature had the external memory for the storage of domain blocks and utilized global data communication for providing domain blocks to all the processors. As the number of processors increases, expanding non-local communication paths is difficult without slowing down the system clock.

In this study, we propose an efficient VLSI architecture for Fixed-size Partitioning Fractal Image Compression (FPFIC), which uses only local communication. The main feature of this architecture is that it is capable of performing the fractal image encoding without the external memory for the fixed domain blocks. We propose the Fast Comparison Module (FCM) in each processor, which can calculate the eight isometric transformations in one full rotation around the center. This module is capable of computing the distortions between the reflected domains and range blocks using the calculation for the rotated domains by the data dependence between domains. We show the validity of

Corresponding Author: B. Sankaragomathi, National Engineering College, Kovilpatti-628 503, Tamil Nadu, India

the operations of FCM. FCM is implemented and laid out in 0.35 μm CMOS technology. The FCM chip contains 334252 transistors in a die size of 2.9×2.9 mm^2. Simulation shows that it has the capability to run up to 40 MHz, or 25 ns per cycle. Finally, we describe the architecture for the larger domain pool.

THE PROPOSED FPFIC ARCHITECTURE

The proposed FPFIC architecture is a systolic array that is arranged in two dimensional space by the same Processor Elements (PEs) that are locally connected and are capable of computing range-domain distortions. Each range block preloaded to each PE is compared with a single domain block in parallel. After the comparison process, all the domain blocks are shifted and the process is repeated. Fast-search is used in sequential fractal image encoding to reduce the number of range-domain comparisons but it was not chosen in this work because it would require random movements of range or domain blocks across the processor array and is difficult to implement this architecture by VLSI technology.

The use of the large domain pool enables more accurate quantization of subtrees of the image, but it results in an increase of the cost of storing the quantization parameters and computational complexity. In the design of FPFIC, the half-overlapping domain pool is considered at first.

Assume that there are $N_p \times N_p$ PEs. Since we assume the total number of PEs is equal to the number of range blocks, $N_p = N_R$. The total number of domain blocks is $(N_p-1)^2$ because the overlapping interval is D/2 = R. Notice that the size of the domain pool D is smaller than the size of the range pool R.

Figure 1 shows an example of the extracting domain pool in the case of $N_R = 4$. In general, a domain block $d_{i,j}$ is drawn from four neighboring range blocks. To save memory, the domain blocks are contracted in advance and are stored in PEs. Let $d_{i,j}$ be a contracted domain block instead of the original domain block. The domain block $d_{i,j}$ is determined as follows:

$$d_{i,j} = E_d(r_{i,j}, ri_{+1,j}, r_{i,j+1}, r_{i+1,j+1}) \quad (1)$$

where E_d operator maps four range blocks to a contracted domain block by averaging (or sub-sampling). The size of the domain block is the same as the range block. The total number of domain blocks extracted from the range pool is (4-1)×(4-1) = 9.

We assume that each range block is loaded into each PE. The encoding procedure of the proposed architecture consists of two stages.

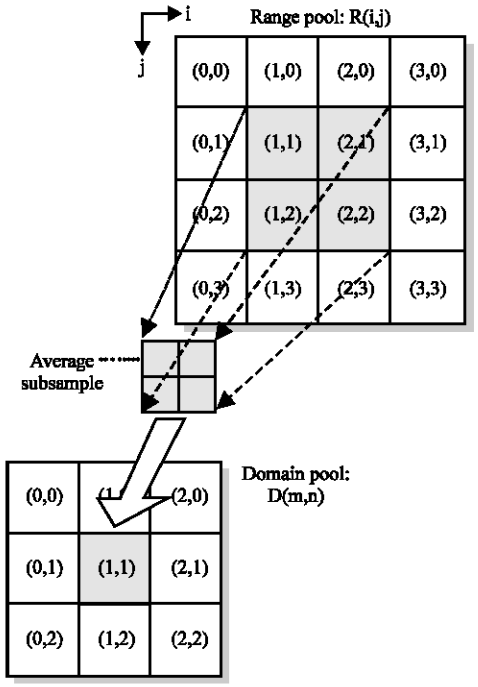

Fig. 1: The extraction of the half-overlapping domain pool

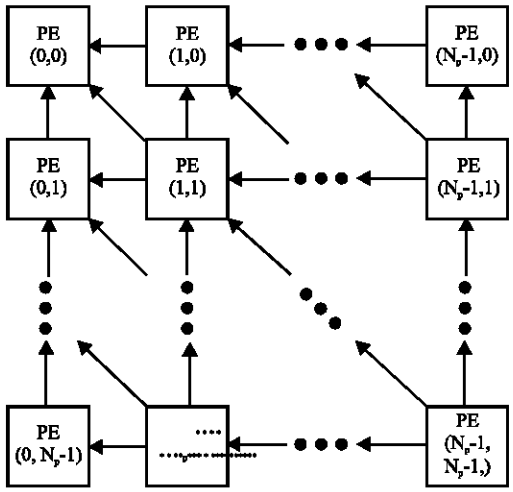

Fig. 2: The proposed architecture for the first stage in encoding procedure

The extraction of the domain pool: stage I: In the first stage, the domain pool is extracted from the range blocks. Figure 2 shows the connection of PEs during the first stage. The directions of the data flow in each PE are determined by data dependence obtained from Eq. (1). In the literature, the architectures have special memory modules to store the domain pool and memory bandwidth becomes a bottleneck, since all PEs access the same

memory to receive domain blocks. In this research, we propose a systolic array which resolves this problem by using only local data communication. The contractive mapping module maps a range block to the quarter-size domain block by averaging (or sub-sampling) and then sends the quarter-size domain block to its three neighboring PEs and domain block memory of itself.

Determining the best domain block: stage II: The second stage determines the best domain block of each range block. The total number of comparisons between the range pool and the domain pool is $(N_R)^2$-$(N_D)^2$×8. In this architecture, each range block is compared with a single domain block in parallel and all domain blocks are shifted to the next PEs. Each PE has a range block and computes the distortion without shifting a domain block to the other PEs by ring connection. $(N_R)^2$×8 steps are needed to search the best matching domain block, where there are eight isometric transformations. diagonal PPs, denoted by the thick-lines, during the distortion calculation.

Each column of PPs computes the sum of pixel differences and adds the sum to the input from the left column of PPs. The results are shifted to the right column of PPs simultaneously. The right-most column of PPs sends the sums of pixel differences from top to bottom. 2C steps are needed for this calculation. The total sum of pixel differences is fed to the four buffers which are used to store the distortion results for the four rotated domain blocks. The results of the distortion calculation for the reflected domain blocks are obtained during the rotation process. The calculation for the reflected domains can be performed on step rotation; the intermediate sums of the distortion for the reected domain blocks are added to the distortion for itself and are shifted to the next PPs.

For executing the eight isometric transformations and selecting the best transformation among them, (360 degree rotation)+(four rotated domains)+(four Reflected domains)+(comparison) = 12C+C/2+3 steps are needed. This module is capable of performing the fast rotations using only one domain block without the external memory for all the domain blocks.

Validity of fast comparison operation: Let $P_{k,l}(t)$ represent the position of a pixel (k, l) of D_1 by counterclockwise step-by-step rotation along the solid-line arrows at time t, where -C+1 ≤ k, l ≤ C-1. The index mapping function M_f of isometric transformation is defined by

$$M_l = \begin{pmatrix} \cos\frac{\pi}{2}\alpha & -\sin\frac{\pi}{2}\alpha \\ \sin\frac{\pi}{2}\alpha & \cos\frac{\pi}{2}\alpha \end{pmatrix} \begin{bmatrix} \beta & 0 \\ 0 & 1 \end{bmatrix} \quad (2)$$

Since M_f maps a pixel (k; l) of the isometric transformed domain to the corresponding position of a range pixel, $T_l(d)(k,l) = d(M_l^{-1}(k,l))$ Thus, we obtain

$$\sum_{k,l} |T_l(d)(k,l) - r(k,l)| = \sum_{kl} |d(k,l) - r(M_l(k,l))| \quad (3)$$

A domain pixel (k, l) transfers to PP $P_{k,l}(t)$ which corresponds to the position of a range pixel, so that PP computes $|r(P_{k,l}(t))-d(k, l)|$.

If $P_{k,l}(t) = M_f(k,l)$, $|r(M_f(k,l))-d(k, l)|$ is computed at PP M_f (k, l) at time t.

By the connection of counterclockwise rotation, four isometric transformations M_f are clearly realized at the time C/2 (f-1). Namely, we have

$$P_{k,l}\left(\frac{C}{2}d\right) = M_{d+1}\begin{pmatrix} k \\ 1 \end{pmatrix} = \begin{pmatrix} K_d \\ l_d \end{pmatrix}, \quad d = 0,1,2,3 \quad (4)$$

The isometric transformations M_f (f = 5,6,7,8) with reflection are not realized at a fixed time, but some particular parts of D_{5-8} which are results of M_f (D_1) are coincided with the shifting movement at some points in time. We will find the time t and the pixel M_f (k,l)(f=5,6,7,8)of D_{5-8} such that $P_{k,l}(t) = M_f(k, l)$.

For C/2≤t <c/2(d+1), t can be represented as t = C/2 d+Δ, where Δ = 0,1, ….., C/2 -1. $P_{k,l}$ (C/2 d+Δ) can be calculated by setting (k_d, l_d) to the base position and moving the positions in the directions as follows:

IF($k_d > 0$, $l_d > 0$) THEN (-1,1) : the first quadrant
IF($k_d < 0$, $l_d > 0$) THEN (-1,-1) : the second quadrant
IF($k_d < 0$, $l_d < 0$) THEN (1,-1) : the third quadrant
IF($k_d > 0$, $l_d < 0$) THEN (1,1) : the fourth quadrant

These directions can be represented by (-sign (l_d), sign (k_d)), where

$$\text{Sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \quad (5)$$

And the moved position is represented as follows:

$$\begin{pmatrix} K_d(\Delta) \\ L_d(\Delta) \end{pmatrix} = \begin{pmatrix} k_d \\ l_d \end{pmatrix} + 2\Delta\begin{pmatrix} -\text{sign}(l_d) \\ \text{sign}(k_d) \end{pmatrix}. \quad (6)$$

Since the connection is restricted, the position ($K_d(\Delta)$, $L_d(\Delta)$) may be outside of the considered quadrant. For that case, the position $P_{k,l}$(C/2 d+Δ) is calculated with (k_{d+1},l_{d+1}) as the base position, since $P_{k,l}$ (C/2 d +Δ) = $P_{k,l}$ (C/2 d+1)-(C/2 -Δ)). By the same discussion for Eq. 6,

we have

$$\begin{pmatrix} K'_{d+1}(\Delta) \\ L'_{d+1}(\Delta) \end{pmatrix} = \begin{pmatrix} K_{d+1} \\ l_{d+1} \end{pmatrix} - (C - 2\Delta)\begin{pmatrix} -\text{sign}(l_{d+1}) \\ \text{sign}(k_{d+1}) \end{pmatrix}. \quad (7)$$

Since the position $P_{k,l}\left(\frac{C}{2}d + \Delta\right)$ is in the same quadrant of (k_d, l_d) or (k_{d+1}, l_{d+1}), the position of pixel at the time t = C/2 d+Δ (d = 0,1,2,3,Δ = 0,1,…, C/2 -1) can be written as

$$P_{k,l}\left(\frac{C}{2}d + \Delta\right) = \begin{cases} \begin{pmatrix} K_d(\Delta) \\ L_d(\Delta) \\ K'_{d+1}(\Delta) \\ L'_{d+1}(\Delta) \end{pmatrix} & \text{if } K_d(\Delta)K_d > 0, \\ & L_d(\Delta)l_d > 0 \\ & \text{if } K'_{d+1}(\Delta)K_{d+1} > 0. \\ & L'_{d+1}(\Delta)l_{d+1} > 0 \\ \text{Delay units} & \text{Otherwise} \end{cases} \quad (8)$$

We calculate the pixel of D_1 rotated to the position of D_5 for the case of d = 0, 1, 2, 3.
From Eq. (8),

$$\begin{pmatrix} -k \\ 1 \end{pmatrix} = \begin{pmatrix} K_o(\Delta) \\ L_o(\Delta) \end{pmatrix} = \begin{pmatrix} K_o \\ L_o \end{pmatrix} + 2\Delta\begin{pmatrix} -\text{sign}(l_o) \\ \text{sign}(k_o) \end{pmatrix} = \begin{pmatrix} k \\ 1 \end{pmatrix} + \begin{pmatrix} -2\Delta \\ 2\Delta \end{pmatrix}. \quad (9)$$

Equation 9 has no feasible solutions for odd k; l > 0 and consider the second case of Eq. 8.

$$\begin{pmatrix} -K \\ 1 \end{pmatrix} = \begin{pmatrix} K'_1(\Delta) \\ L'_1(\Delta) \end{pmatrix} = \begin{pmatrix} K_1 \\ L_1 \end{pmatrix} - (C - 2\Delta)\begin{pmatrix} -\text{sign}(l_1) \\ \text{sign}(k_1) \end{pmatrix} = \begin{pmatrix} -l \\ k \end{pmatrix} - \begin{pmatrix} -C + 2\Delta \\ -C + 2\Delta \end{pmatrix}. \quad (10)$$

Therefore

$$l = k + C - 2\Delta \quad (11)$$

Hence

$$\Delta = \frac{1}{2}(k - l + C), t = \frac{C}{2}.0 + \Delta = \frac{1}{2}(k - l + C) \quad (12)$$

The conditions for the second case of Eq. 4.8:

$$\begin{aligned} K'_1(\Delta)k_1 &= (-l + C - 2\Delta)(-l) > 0 \\ L'_1(\Delta)l_1 &= (K + C + 2\Delta)(K) > 0 \end{aligned} \quad (13)$$

Give a range of (k; l) such that

$$\begin{cases} l > C - 2\Delta. \\ K > 0 \end{cases} \quad (14)$$

From the first case of Eq. 8,

$$\begin{pmatrix} -k \\ 1 \end{pmatrix} = \begin{pmatrix} k_1 \\ l_1 \end{pmatrix} + 2\begin{pmatrix} -\text{Sign}(L_1) \\ \text{Sign}(K_1) \end{pmatrix}\Delta = \begin{pmatrix} -L \\ k \end{pmatrix} + \begin{pmatrix} -2\Delta \\ -2\Delta \end{pmatrix}. \quad (15)$$

Thus,

$$l = k - 2\Delta \quad (16)$$

Hence

$$\Delta = \frac{1}{2}(k - l), t = \frac{C}{2}, l + \Delta = \frac{1}{2}(k - l + C). \quad (17)$$

The conditions for the first case of Eq. 8:

$$\begin{cases} (-l - 2\Delta)(-l) > 0 \\ (k - 2\Delta)(k) > 0 \end{cases} \quad (18)$$

Given a range of (k; l) such that

$$\begin{aligned} l &> 0 \\ k &> 2\Delta \end{aligned} \quad (19)$$

The time at which the computation with the pixels (k, l) and $(\hat{k}, \hat{l})$ is computed can be written from Eq. 12 and 17 as follows:

$$l = \frac{1}{2}(k - l + C) \quad \text{if } k,l > 0 \quad (20)$$

RESULTS AND DISCUSSION

Each PE has the Extended Right Domain Memory (ERDM), the Extended Bottom Domain Memory (EBDM) and the Extended Diagonal Domain Memory (EDDM) to store the extended domain data from other PEs as follows:

$$\begin{aligned} d_{ERDM(i,j)} &= E_d(r_{(i+2,j),r(i+2,j+1)}) \\ d_{EBDM(i,j)} &= E_d(r_{(i,j+2),r(i+1,j+2)}) \\ d_{EDDM(i,j)} &= E_d(r_{(i+2,j+2)}) \end{aligned} \quad (22)$$

where E_d is sub-sampling operator.

Table 1: Estimated encoding time

Image dimension [Pixel]	Cycles	Encoding time [see]
256×256	217088	0.0054
512×512	868352	0.0217

These groups of data are transmitted immediately from the three neighboring PEs after the first stage. At the end of the comparison using the half-overlapping domain pool, ERDM, EBDM and EDDM send a column and/or a row line of data to its domain memory. The encoding procedure is repeated until all range blocks have compared themselves against every domain block. The number of steps needed for the comparisons using D_P is $(R/_2)^2 \times (N_D)^2$, which is larger than $(N_{DP})^2$ since the right-most column and bottom PPs include incomplete domain blocks by moving of domain data. This FPFIC architecture is very modular and hierarchical due that FPFIC consists of PE array and each PE has many PPs. Since the FCM of the PE is the most important module, it has been considered in this experiment. We implement a FCM in Verilog HDL to evaluate the features of the architecture. Since MAD is suitable for VLSI implementation with the aim of cutting down on computation time and/or area, MAD is used as the distortion criterion in this experiment.

The FCM chip contains 334252 transistors in a die size of 2.9×2.9 mm^2. Simulation shows that it has the capability to run up to 40 MHz, or 25ns per cycle. If every operation is performed in one clock cycle, the number of clock cycles required for encoding one image will be: $(12C+C/2+3)\times(N_p)2$, where $(N_p)^2$ FCM chips are contained in the FPFIC system. Table 1 gives encoding time estimations based on cycles per operation. The estimation result shows that the FPFIC system encodes 256×256 and 512×512 images in 5.4 and 21.7 ms, respectively, where the size of range block is 4×4 image and 40 MHz clock signal. If the frame rate is 30 frames per second, encoding one frame should be processed in 33.3 ms. Therefore, the FPFIC system with $(Np)^2$ FCM chips can meet the real time video applications.

The other parts of PE consist of memory module, contractive mapping module, Mux and Dec, which are required small size circuits. If one PE can be implemented in a chip, FIC system must be implemented with many chips in order to encode the image. The number of chips needed for FIC system is the total number of range blocks, $N_R \times N_R$.

We have proposed the Fast Comparison Module (FCM) in each processor which can calculate the eight isometric transformations in one full rotation around the center and have shown the validity of the algorithm of FCM. It comes out that by using FCM gains a speed-up of 2 times over the general pipeline processing.

CONCLUSION

We have proposed an efficient VLSI architecture for Fixed-size Partitioning Fractal Image Compression (FPFIC), which uses only local communication. The main feature of this architecture is that it is capable of performing the fractal image encoding without the external memory for the fixed domain blocks. The proposed Fast Comparison Module (FCM) in each processor which can calculate the eight isometric transformations by one full rotation around the center. This module is capable of computing the distortions between the reflected domains and range blocks using the results of the distortion calculation for the rotated domains by the data dependence between domains. The FPFIC architecture for the P-pixels-interval domain pool has been proposed. Each PE of this architecture has the Extended Right Domain Memory (ERDM), the Extended Bottom Domain Memory (EBDM) and the Extended Diagonal Domain Memory (EDDM) to store the extended domain data from other PEs. We have implemented FCM with ROHM 0.35 μm CMOS technology. The FCM chip contains 334252 transistors in a die size of 2.9×2.9 mm^2. Simulation shows that it has the capability to run up to 40 MHz, or 25 ns per cycle. If one PE can be implemented in a chip, FIC system must be implemented with NR×NR chips.

REFERENCES

Acken, K.P., H.N. Kim, K.J. Irwin and R.M. Owens, 1996. An Architectural Design for Parallel Fractal Compression, Proceedings International Conference on Application-Specific System, Architectures and Processors, pp: 3-11.

Acken, K.P., M.J. Irwin and R.M. Owens, 1998. A Parallel ASIC Architecture for Efficient Fractal Image Coding. J. VLSI Signal Processing System. Signal Image Video Tech., pp: 97-113.

Ancarani, F., A. De Gloria, M. Olivieri and C. Stazzone, 1996. Design of an ASIC Architecture for High Speed Fractal Image Compression, Proceedings 9th Ann. IEEE. Int. ASIC Conf. Exhibit, pp: 223-226.

Barnsley, M.F., 1988. Fractals Everywhere, Academic Press Inc., San Diego.

Davoine, F., J. Svensson and J.M. Chassery, 1995. A Mixed Triangular and Quadrilateral Partition for Fractal Image Coding, IEEE. Int. Conf. Image Processing (ICIP).

Fatemi, O. and S. Panchanathan, 1997. Fractal Engine, Proc. SPIE Int. Soc. Optical Eng, pp: 88-99.

Fisher, Y., 1995. Fractal Image Compression Theory and Application, Springer Verlag, New York.

Foster, M.J. and H.T. Kung, 1980. The design of special-purpose VLSI chips, IEEE. Computers, pp: 26-40.

Ida, T. and Y. Sambonsugi, 1998. Image segmentation and contour detection using fractal coding, IEEE. Trans. Circuits and Systems for Video Tech., pp: 968-975.

Jacobs, E.W., Y. Fisher and R.D. Boss, 1992. Image Compression: A Study of the Iterated Transform Method, Signal Process, pp: 251-263.

Jacquin, 1993. Fractal image coding: A Rev., Proc. IEEE., pp: 1451-1465.

Jacquin, A., 1992. Image coding based on a fractal theory of iterated contractive image transformations, IEEE. Trans. Image Processing, pp: 18-30.

Lafruit, G., F. Catthoor and J.P.H. Cornelis, 1999. An Efficient VLSI Architecture for 2-D Wavelet Image Coding with Novel Image Scan, IEEE. Trans. VLSI Sys., pp: 56-68.

Meal, C.A. and L.A. Conway, 1980. Introduction to VLSI systems, Addison-Wesley, Reading, Mass.

Reusens, E., 1994. Overlapped adaptive partitioning for image coding based on the theory of iterated function systems, Proc. of ICASSP, pp: 569-572 .

Zhao, Y. and B. Yuan, 1998. A new affine transformation: Its Theory and Application to Image Coding. IEEE. Trans. Circuits and Systems for Video Tech., pp: 269-274.