設 計 自 動 化 89－9
(1998. 9. 22)

フラクタル画像圧縮のVLSIアーキテクチャ

○ 李 信行, 阿曽 弘具

東北大学 工学部 電気・通信工学科
〒980-8579 仙台市青葉区荒巻字青葉
TEL 022-217-7088　　E-mail lee@aso.ecei.tohoku.ac.jp

あらまし　フラクタル画像圧縮は高速探索用の電子図書館の画像蓄積などに有効な画像圧縮法として注目されている．この手法は符号化時間が膨大になる欠点があり，そのためフラクタル専用ハードウェアが必要である．本稿ではフラクタル画像圧縮のVLSIアーキテクチャを提案する．まず各プロセッサの持つレンジブロックからドメインブロックを生成し，ドメインブロックを他のプロセッサに規則的なデータ通信を行なうことで符号化する．

キーワード　フラクタル画像圧縮，VLSIアーキテクチャ

A VLSI Architecture for Fractal Image Coding

○ Shinhaeng Lee and Hirotomo Aso

Department of Electrical and Communication Engineering, Faculty of Engeneering,
Tohoku University,
Aoba, Aramaki, Aoba-ku, Sendai, 980-8579, Japan
TEL (+81)22-217-7088　　E-mail lee@aso.ecei.tohoku.ac.jp

Abstract　Fractal image compression has received great attention in image storage of electronic library for a quick access. The main problem of fractal image compression is a long encoding time. For this reasion, the dedicated ASIC architecture for fractal image coding is needed. In this paper, we propose an efficient VLSI architecture for fractal image coding. First, domain blocks are formed from the range blocks in processors and the encoding procedure is performed by the regular data flow of domain blocks into the other processors.

key words　fractal image compression, VLSI architecture

1. Introduction

Image compression has been important in relation to image storage and image transmission. Fractal image coding, proposed by Bansley and realized by Jacquin [1][2][3], has received great attention in digital image compression [4]. The principle of fractal image coding is that the image converges to a stable image by iterating the contractive transformations on an arbitrary initial image. The image is partitioned into a number of blocks and is coded by obtaining only the information of the contractive transformations for the blocks.

Similarly, the main problem of fractal image coding is the tremendous encoding time needed due to the large amount of comparisons between domain and range blocks. A few dedicated architectures proposed have been utilized global data communication to providing domain blocks to all the processors [5][6][7][8]. As the number of processors increase, expanding non-local communication paths is difficult without slowing down the system clock.

In this paper, we propose an efficient VLSI architecture using only local communication such that each processor has memory for a range and a domain block which is shifted to the next processors.

2. Review of Fractal Image Coding

In this section, we will present a review of the principle of fractal image coding.

2.1 Theoretical Foundations

Let (M, δ) denote a metric space of digital image, where M is the space of discrete domains finitely bounded, and δ is a given distortion measure. A transformation τ on a metric space (M, δ) is contractive if there is a constant $0 \leq s < 1$ such that

$$\delta(\tau(\mu), \tau(\nu)) \leq s\delta(\mu, \nu), \quad \forall \mu, \nu \in M. \tag{1}$$

The factor s is called the *contractivity factor* of τ.

If M is a complete metric space, then there exists a fixed point μ_a for τ, called *attractor*, such that

$$\tau(\mu_a) = \mu_a, \quad \mu_a \in M. \tag{2}$$

Moreover, if τ is contractive then τ may be iteratively applied to any image μ_0, yielding a stable image μ_a [1]:

$$\lim_{n\to\infty} \tau^n(\mu_0) = \mu_a, \quad \forall \mu_0 \in M, \tag{3}$$

where τ^n means the n^{th} iterate of τ.

Let μ_{orig} be an original image we want to encode, and constractive such that $\tau(\mu_{orig}) = \mu_{orig}$. Then the transformation τ can be seen as a *lossy image code* for μ_{orig}.

2.2 Fractal Coding Algorithm

Provided μ_{orig} is an $N \times N$ pixels gray scale image, we partition the original image μ_{orig} into a set of non-overlapping $R \times R$ pixels *range blocks* $\{r_{(i,j)}\}$, as follows:

$$\mu_{orig} = \bigcup_{i,j=1}^{N_R} r_{(i,j)}, \quad r_{(i,j)} \bigcap r_{(\hat{i},\hat{j})} = \emptyset \text{ for } (i,j) \neq (\hat{i},\hat{j}), \tag{4}$$

where $r_{(i,j)}$ represents the range block at coordinates (i, j), $N_R \times N_R$ is the total number of range blocks. $\mathcal{R} = \{r_{(i,j)} | 1 \leq i, j \leq N_R\}$ is called *range pool*. A set of overlapping $D \times D$ $(D = 2R)$ pixels *domain blocks*, $\{d_{(m,n)} | 1 \leq m, n \leq N_D\}$, are drawn from *domain pool* $\mathcal{D}$ where $N_D \times N_D$ is the total number of domain blocks.

A variety of domain pools is used in the literature. Fixed-spacing domain pools are the *two-pixels-interval domain pool* $\mathcal{D}_{two}$ and *half-overlapping domain pool* $\mathcal{D}_{half}$, which are overlapped along the vertical and horizontal directions with the overlapping interval set to 2 pixels and $D/2$ pixels, respectively.

For each range block, the coding procedure is to find one domain block and its transformation which is the best match of the original range block. The mapping for the $(i,j)^{th}$ range block, $\tau_{(i,j)}$, consists of a scaling factor $s_{(i,j)}$, offset $o_{(i,j)}$, pixel isometry $t_k, 1 \leq k \leq 8$, and spatial contraction S. A pixel isometry t_k maps a square block to one of the 8 isometries obtained from compositions of reflections and 90 degree rotations.

The result of applying this mapping is an approximation to the $(i,j)^{th}$ range block, $\tilde{r}_{(i,j)}$ as

follows:

$$\tilde{r}_{(i,j)} = s_{(i,j)} t_{I(i,j)}(S(d_{N(i,j)})) + o_{(i,j)}, \quad (5)$$

where $N(i,j)$ is a domain block selection function, which associates the $(i,j)^{th}$ range block with a domain block from $\mathcal{D}$ and $I(i,j)$ is an isometry selection function, which maps the $(i,j)^{th}$ range block to one of a set of possible isometry operations.

The encoding process is determining the parameters in equation (5) for all range blocks such that the distortion between each range block and its approximation, $\delta(\tilde{r}_{(i,j)}, r_{(i,j)})$, is minimized.

The fractal image encoding procedure is shown as follows:

STEP 1:

Extract range pool $\mathcal{R}$

STEP 2:

Extract domain pool $\mathcal{D}$

STEP 3:

For each range block

For each domain block and each isometry

Compute distance between blocks

Determine the best domain block

STEP 4:

Consider the parameters of the best as a code

In the encoding phase, the most computationally intensive operations are the computation of distance between range and transformed domain blocks (STEP 3). The total number of computations of distance is $(N_R) \times (N_D)^2 \times 8$.

3. The Proposed Architecture for Fractal Image Encoding

We propose an efficient systolic architecture for fractal image encoding. The proposed systolic array is arranged in two dimensional space by the same processor elements(PE's) that are locally connected and are capable of computing a range-domain distance. Each range block pre-loaded to each PE is compared with a single domain block in parallel. All the domain blocks are shifted and the process is repeated.

There are two searching schemes, one is *full-search* and the other is *fast-search* by classification. The full-search scheme can produce better performance than fast-search scheme and possesses a very regular data flow. Fast-search is used in sequential fractal encoding to reduce the number of range-domain comparisons but it was not chosen in this paper because it would required random movement of range or domain blocks across the processor array.

In the design of systolic array to implement the fractal encoding algorithm in this paper, both of half-overlapping domain pool and two-pixels-interval domain pool are considered.

3.1 Half-overlapping Domain Pool

Assume that there are $N_p \times N_p$ PE's. Since we assume the total number of PE's is equal to the number of range blocks, $N_p = N_R$. Now the domain pool is a half-overlapping one $\mathcal{D}_{half}$. The total number of domain blocks is $(N_p - 1)(N_p - 1)$ due that the overlapping interval is $D/2 = R$. Notice that the size of domain pool $\mathcal{D}_{half}$ is smaller than the size of range pool $\mathcal{R}$.

Figure 1 shows an example of extracting domain pool where N_R is 4. In general, domain block (i,j) is drawn from 4 neighbor range blocks as follows:

$$d_{(i,j)} = E_d(r_{(i,j)}, r_{(i+1,j)}, r_{(i,j+1)}, r_{(i+1,j+1)}) \quad (6)$$

where E_d operator maps a $D \times D$ pixels block to a $R \times R$ pixels block by averaging each pixel with its neighbors and then sub-sampling. The number of total domain blocks extracted from range pool is $(4-1)(4-1) = 9$ in Figure 1.

We assume that each range block is loaded into each PE. The encoding procedure of the proposed architecture consist of two phases. In the first phase, domain pool is extracted. Figure 2 shows the connection of PE's in first phase. The direction of the data flow in each PE is determined by data dependency obtained from equation (6).

The structure of PE of the proposed architecture is shown in Figure 3. Each PE has three memory modules, an isometry function module, an average-and-subsample module, a decoder, and a distance-computation module. Three memory modules consist of a range block memory, a domain block memory, and a code memory to

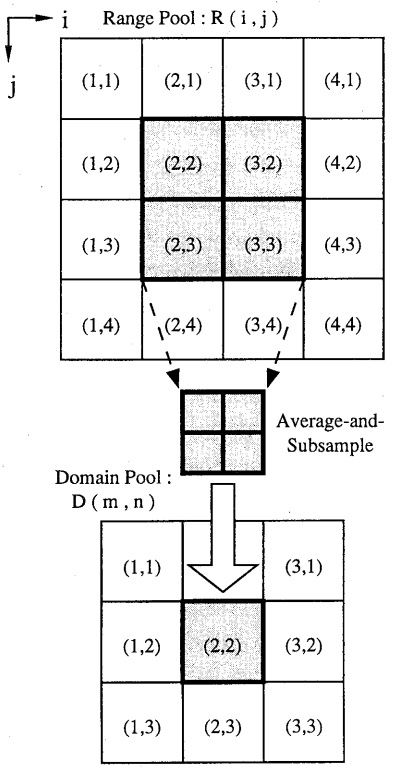

Figure 1. Extracting domain pool.

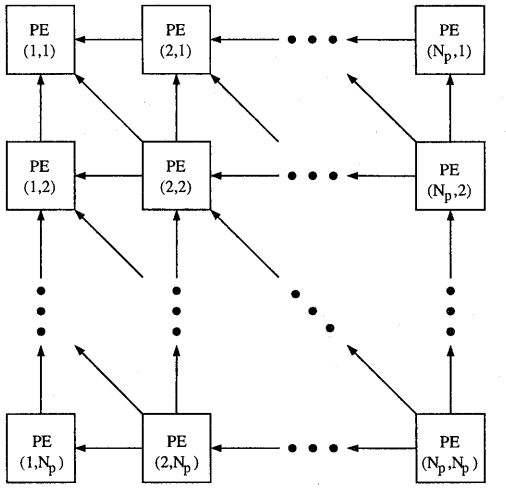

Figure 2. The proposed architecture for first phase in encoding procedure.

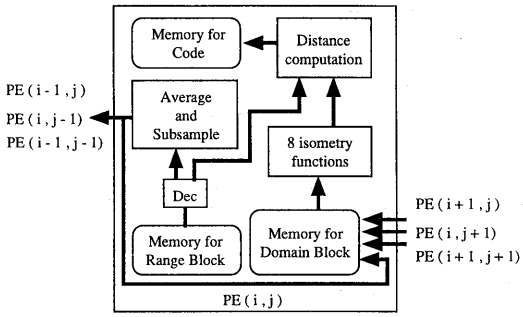

Figure 3. The structure of PE of the proposed architecture.

store results obtained. Each range block is loaded into each range block memory. In the literature, the architectures have the special memory module to store domain pool [6][7] and memory bandwidth becomes a bottleneck since all PE's access the same memory to receive domain blocks. However, the proposed systolic architecture resolves this problem by using only local data communication. In the proposed architecture, each PE extracts a quater of a domain block from a corresponding range block and stores it into domain block memory.

The average-and-subsample module maps a range block to the half-size domain block by averaging each pixel with its neighbors and then sub-sampling and send the half-size domain block to three neighbor PE's and domain block memory of itself. The isometry function module executes eight isometry transformations on its domain block. The distance-computation module calculates the distance between a range block and domain blocks.

The second phase computes distance between range and domain blocks. The number of comparisons between range pool and domain pool is $(N_R)^2 \times (N_D)^2$. In this architecture, each range block is compared with a single domain block in parallel and all domain blocks are shifted to be compared with the next range block. Note that the systolic computation of the comparison requires a topology of the linear array i.e. a *ring connection*. For the systolic computation using a simple linear array, $N_R \times N_R$ steps are needed.

To improve parallelism, we propose a new connection of PE's by modifying a ring connection.

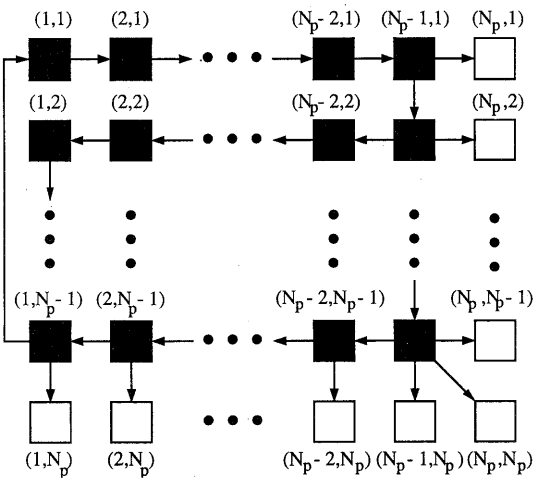

Figure 5. The connection of PE's for computation of the comparison.

Figure 5 shows the connection of PE's, where N_p is an odd number. If N_p is an even number, the path from PE ($N_p - 1$,$N_p - 1$) to (1,1) is needed instead of the path from PE (1,$N_p - 1$) to (1,1) in Figure 5. In Figure 5, each black square has a range block and a domain block which is shifted at the next step along with the connection. Each white square has a only range block and computing the distance without shifting a domain block to the other PE's.

The data flow of the proposed architecture is shown in Figure 4 where $N_R = 4$. In step 0 in Figure 4, nine domain blocks are extracted from range blocks. Steps 1-9 shows the flow of domain blocks. $N_D \times N_D$ steps are performed for the comparisons by the proposed connection of PE's. Since N_D of $\mathcal{D}_{half}$ is smaller than N_R, the proposed architecture is faster than the straightforward architecture using ring connection.

3.2 Two-pixels-interval Domain Pool

If higher quality images are needed, you have to use the larger domain pool such as two-pixels-interval domain pool $\mathcal{D}_{two}$. The proposed architecture for $\mathcal{D}_{two}$ can be designed by modification of domain block memory as shown in Figure 6. Each PE has the extended right domain memory(ERDM), the extended bottom domain memory(EBDM) and the extended diagonal memory(EDDM) to store the extended domain data from other PE's as follows:

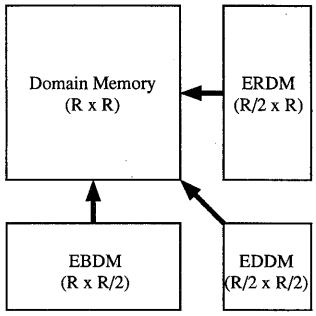

Figure 6. The modified domain block memory for $\mathcal{D}_{two}$.

$$\begin{aligned} d_{ERDM(i,j)} &= E_d(r_{(i+2,j)}, r_{(i+2,j+1)}) \\ d_{EBDM(i,j)} &= E_d(r_{(i,j+2)}, r_{(i+1,j+2)}) \\ d_{EDDM(i,j)} &= E_d(r_{(i+2,j+2)}) \end{aligned} \quad (7)$$

These data are transmitted immediately after the first phase before the second phase from the three neighbor PE's. After the comparison process as soon one for $\mathcal{D}_{half}$, ERDM, EBDM and EDDM send a line of data to its domain memory. Since the data in domain memory is obtained by sub-sampling, a pixel of the extended domain memory is the data of two pixels in range pool. The two-pixels-interval domain pool is extracted and repeat STEP 3 and STEP 4 in the encoding procedure until each range block has compared itself against every domain block. $(\frac{R}{2})^2 \times (N_D)^2$ steps are performed for the comparisons using $\mathcal{D}_{two}$.

4. Conclusions

In this paper, we have proposed an efficient VLSI architecture for fractal image coding. The encoding scheme is based on the fixed-size range blocks and the full-search. Domain pool is formed form the range blocks in processors and the comparison process is performed by the local communication since each PE has the memory for a range and a domain block. This architecture can perform the comparison process in $N_D \times N_D$ steps by the modification of the connection of PE's, where half-overlapping domain pool ($N_D < N_R$).

Steps	PE(1,1)	PE(2,1)	PE(3,1)	PE(4,1)	PE(1,2)	PE(2,2)	PE(3,2)	PE(4,2)
0	D(1,1)	D(2,1)	D(3,1)		D(1,2)	D(2,2)	D(3,2)	
1	D(3,3)	D(1,1)	D(2,1)	D(3,1)	D(2,2)	D(3,2)	D(3,1)	D(3,2)
2	D(2,3)	D(3,3)	D(1,1)	D(2,1)	D(3,2)	D(3,1)	D(2,1)	D(3,1)
3	D(1,3)	D(2,3)	D(3,3)	D(1,1)	D(3,1)	D(2,1)	D(1,1)	D(2,1)
4	D(1,2)	D(1,3)	D(2,3)	D(3,3)	D(2,1)	D(1,1)	D(3,3)	D(1,1)
5	D(2,2)	D(1,2)	D(1,3)	D(2,3)	D(1,1)	D(3,3)	D(2,3)	D(3,3)
6	D(3,2)	D(2,2)	D(1,2)	D(1,3)	D(3,3)	D(2,3)	D(1,3)	D(2,3)
7	D(3,1)	D(3,2)	D(2,2)	D(1,2)	D(2,3)	D(1,3)	D(1,2)	D(1,3)
8	D(2,1)	D(3,1)	D(3,2)	D(2,2)	D(1,3)	D(1,2)	D(2,2)	D(1,2)
9				D(3,2)				D(2,2)
Steps	PE(1,3)	PE(2,3)	PE(3,3)	PE(4,3)	PE(1,4)	PE(2,4)	PE(3,4)	PE(4,4)
0	D(1,3)	D(2,3)	D(3,3)					
1	D(1,2)	D(1,3)	D(2,3)	D(3,3)	D(1,3)	D(2,3)	D(3,3)	D(3,3)
2	D(2,2)	D(1,2)	D(1,3)	D(2,3)	D(1,2)	D(1,3)	D(2,3)	D(2,3)
3	D(3,2)	D(2,2)	D(1,2)	D(1,3)	D(2,2)	D(1,2)	D(1,3)	D(1,3)
4	D(3,1)	D(3,2)	D(2,2)	D(1,2)	D(3,2)	D(2,2)	D(1,2)	D(1,2)
5	D(2,1)	D(3,1)	D(3,2)	D(2,2)	D(3,1)	D(3,2)	D(2,2)	D(2,2)
6	D(1,1)	D(2,1)	D(3,1)	D(3,2)	D(2,1)	D(3,1)	D(3,2)	D(3,2)
7	D(3,3)	D(1,1)	D(2,1)	D(3,1)	D(1,1)	D(2,1)	D(3,1)	D(3,1)
8	D(2,3)	D(3,3)	D(1,1)	D(2,1)	D(3,3)	D(1,1)	D(2,1)	D(2,1)
9				D(1,1)	D(2,3)	D(3,3)	D(1,1)	D(1,1)

Figure 4. The data flow of the proposed architecture.

In addition, we have proposed the modified architecture for higher quality images with the larger domain pool. This can compute the comparisons between range pool and the larger domain pool obtained by the extended domain memory in $(\frac{R}{2})^2 \times (N_D)^2$ steps.

Further research will focus on the architecture for fractal image coding using the flexible-size partition schemes such as the quad-tree partition.

References

[1] M. F. Barnsley, *Fractals Everywhere*, Academic Press Inc. , San Diego, 1988.

[2] A. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Processing*, vol. 1, no. 1, pp. 18-30, Jan. 1992.

[3] A. Jacquin, "Fractal Image Coding: A Review," *Processing of the IEEE*, vol. 81, no. 10, pp. 1451-1465, Oct. 1993.

[4] Y. Fisher, *Fractal Image Compression Theory and Application*, Springer Verlag, New York, 1995.

[5] F. Ancarani, A. De Gloria, M. Olivieri, and C. Stazzone, "Design of an ASIC Architecture for High Speed Fractal Image Compression," *Proceedings Ninth Annual IEEE international ASIC Conference and Exhibit*, pp. 223-6, 1996.

[6] O. Fatemi and S. Panchanathan, "Fractal Engine," *Proc. SPIE Int. Soc. Opt. Eng.* , vol. 3021, pp. 88-99, 1997.

[7] Z. L. He, M. L. Liou, and K. W. Fu, "VLSI Architecture for Real-Time Fractal Video Encoding," *1996 IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 738-41, 1996.

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