Design, Simulation and Synthesis of an ASIC for Fractal Image Coding

S.K. Bhunia, S.K. Ghosh, P. Kumar, P.P. Das, J. Mukherjee
Department of Computer Science and Engg.
Indian Institute of Technology, Kharagpur
Kharagpur -721302, INDIA
som,jay@cse.iitkgp.ernet.in

Abstract

In recent years, fractal encoding of image (using Iterated Function Systems) has emerged as a potent technique in the field of image compression. This method has comparable or sometimes better performance as compared to most others, specially, with respect to quality of reconstructed image and compression ratio. But most fractal encoding methods are very slow that prevent them from realtime processing of images. Since fractal encoding methods have ample data parallelism and spatial/temporal recurrence, there is lot of scope for designing efficient VLSI architectures for them. In this paper, we have selected a suitable encoding scheme and designed the VLSI architecture for it. The proposed architecture has been simulated and synthesized, using Verilog and Synergy of Cadence Design Tools. The architecture employs principles of pipelining and parellelism to enhance performance with respect to speed of compression. Simulation and synthesis results show good time performance for the proposed chip.

1. Introduction

A number of different image compression schemes have been developed till date. They can be broadly classified into two categories : *lossless* or *error-free* compression (like bitplane encoding or variable length encoding) and *lossy compression* (like transform based coding) methods. The currently popular JPEG[1] image compression method is based on Discrete Cosine Transform (DCT) and is now an international standard.

In addition there have been two exciting developments over the past few years : *Wavelet based encoding* and *compression based on Fractal geometry* [5]. In Wavelet based encoding the image data is decorrelated by Wavelet Transform, the resulting transform coefficients are then quantized and coded. On the other hand, Fractal Image Compression (FIC) method is based on the concept of Partitioned Iterated Funtion System(PIFS) [4] [3] that models an image as the attractor of a PIFS. Most of the fractal encoding schemes are inherently computation intensive and results in poor compression speed.

While there has been extensive research to enhance the encoding speed of fractal schemes - mostly based on some 'classification' of image segments [1], there is no known attempt to find an efficient hardware implementation for any fractal encoding scheme. Moreover since most fractal encoding schemes have ample data parallelism inherent in the algorithm there is enough possibility to exploit them to enhance encoding speed and throughput.

Starting with this motivation we have come up with a high-speed parallel VLSI architechure for a fractal encoding method [2]. The architechure exploits the principle of parallelism and pipelining to make the encoding process time efficient. In the course of our work, we initially dealt with a parallel nonpipelined architecture - designing and verifying the same and thereafter modified it for pipelined implementation keeping the parallelism intact. The pipelined architecture gave considerable speedup with little increase in hardware complexity.

The architechure is designed to be implemented in a single chip. It is a special purpose chip or ASIC (Application Specific Integrated Circuit) and has the distinct advantages of simplicity, lower production cost and higher reliability.

Both the architectures have been simulated using Verilog Hardware Description Language (HDL) and synthesized using Synergy. The simulation results show a significant improvement in speed of encoding. We believe, a more efficient architecture for fractal encoding comparable in performance to the available JPEG chips will be possible in future.

[1] Joint Photographic Expert Group

0-7695-0013-7/99 $10.00 © 1999 IEEE

2. Overview of Fractal Image Encoding

In fractal compression an image is encoded as the attractor of an iterated function system. It is based on the observation that natural images are partially self-transformable [4]. They contain 'affine redundancy' in the sense that a block in the image (called *range*) can be derived from another block of the same image (called *domain*) by some affine transformation. The image is encoded as the union of best-fitting affine transformation and the corresponding image domain blocks for all segments consisting of the image support. In all fractal encoding methods the encoding process starts with partitioning of the image into a set of non-overlapping segments(*range blocks*) and then for each range block an image block (*domain block*) with different resolution is searched that gives the best affine mapping to the range segment. Compression is achieved by encoding the domain and the affine transformation for each range block. The speed of compression primarily depends on the efficiency of search for suitable *domain* for a *range block*. Here we have adopted an *ASIC Design* for speedy compression using FIC by suitably casting it into a pipelined parallel hardware architecture.

Before discussing the ASIC Design, a brief overview of the basic algorithm is given below:

2.1. The Base Algorithm

Algorithm Base FIC
Input: A $2^n \times 2^n$ Image.
Output: Stream of fractal coded doublets.

1. Partition the image into a set of non-overlapping square range (R) blocks, each of size R_SizeXR_Size and containing $N = R\_Size^2$ pixels. Mark all the ranges in R *uncovered.*
2. Partition the image into a set of overlapping square domain (D) blocks and each of size double that of a range with lattice spacing l.
3. While there is *uncovered* range R_i in R do
   (a) Pick the range R_i and mark it *covered.*
   (b) Mark all the domains in D *unexplored.*
   (c) While there is *unexplored* domain D_j in D
      i. Pick the domain D_j and mark it *explored.*
      ii. Shrink the domain to the size of range by averaging four (2x2) adjacent pixel values.
      iii. Perform eight spatial transformations of the domain. These are identity, rotation by 90°, 180° and 270° and flips about horizontal, vertical, diagonal and antidiagonal axes.
      iv. For K_d = $Lower$ to $Upper$ do
         A. For all spatial transformations, Compute

$$K = K_d * Step\_Size \quad (1)$$

$$E = \frac{1}{N}\sum_{k=1}^{N}[r_k - (d_k + K)]^2 \quad (2)$$

where $r_k \in R_i, d_k \in$ Transformed D_j

         B. If ($E < Thresh$) store the domain, the quantized value of K (that is, K_d), and the spatial transformation in the output.
         Go to Step 3. /* A match for the present range is found. Goto next range.*/
         else continue (goto(c)). /* Pick up next domain D_j */

   (d) If after all domains in D are *explored* and none with $E < Thresh$ is found then, store the domain, K_d and the transformation which give the minimum error.

2.1.1. Optimum Value of K and Error Computation

The K is grey value offset quantized in steps of $Step\_Size$. The expression for optimum value of K can be obtained by setting $\frac{dE}{dK} = 0$, as :

$$K_o = \frac{1}{N}[\sum_{i=1}^{N} r_i - \sum_{i=1}^{N} d_i] \quad (3)$$

Note that d_i's here are dependent on the spatial transformation used, yet $\sum_i d_i$ is an invariant. Therefore optimal value of K is the same for all spatial transforms. The error now turns out to be as follows:

$$Term1 = \sum_{i=1}^{N}(r_i - d_i)^2 \quad (4)$$

$$Term2 = K_o * (K_o * N - 2 * (\sum_{i=1}^{N} r_i - \sum_{i=1}^{N} d_i)) \quad (5)$$

$$E = (Term1 + Term2)/N \quad (6)$$

K_o is quantized by dividing it by $Step\_Size$.

In the modified algorithm we compute $Term2$ once for all spatial transformation s after computing the optimal offset K_o and for each transformation we compute $Term1$ separately.

3. The VLSI Architecture

The overall system architecture of the ASIC for the selected fractal encoding scheme is given in Fig. 1. The hardware organization shown in the architecture reflects the sequence

of computations in the encoding process. The critical part of this computation is the strategy for domain access, performing eight transformation in parallel and error computation.

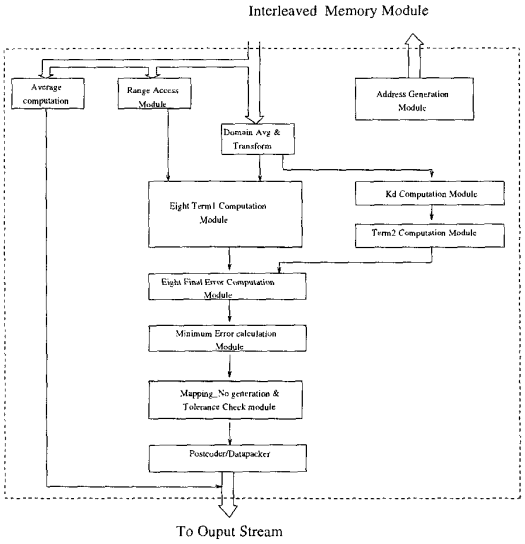

Figure 1. The Overall System Structure

3.1. Details of Implementation

The block diagram of the whole ASIC is shown in Fig. 2. Most of the modules have been implemented in the gate level. The memory containing the image is external to the chip. The control unit has not been shown in the system architecture.

3.1.1. Address Generation Module (A)

Fig. 2 shows this module with its three submodules : one for generating address for the computation of initial image averages (*Av-Addr*), one for address generation for the range pixels (*R-Addr*), and the other for generating the domain pixel addresses (*D-Addr*).

3.1.2. Initial Image Computation Module (B)

The computation of averages of fixed size blocks is done in following three sequential steps (Fig. 2) : (i) a high speed adder is used to sum all the pixel values of the block, (ii) the average is determined by shifting the sum right by appropriate distance. (iii) the average value is quantized by right shifting the average value.

3.1.3. Range Access Module (C)

This module (Fig. 2) performs two simple tasks: i) storing the range pixels in registers and ii) computing the summation of the pixel values.

3.1.4. Domain Average and Transformation Module (D)

This module consists of three major submodules (Fig. 2). The first submodule computes the average value of each 2x2 subsquare. It has again two components - an adder and a shift register. The second submodule consists of the logic required to compute $\sum_{i=1} d_i$. Since the domains overlap, while accessing a new domain only the portion of it which does not overlap with the previous domain need to be fetched, reducing the number of memory accesses. The third submodule of the domain average and transformation module consists of the eight spatial transformation units.

3.1.5. Error Computation Module

This module does computation work and comprises of five submodules (Fig. 2), namely, K_d Computation Module (E), $Term2$ Computation Module (F), $Term1$ Computation Module (G), Min_Term1 Computation Module (H) and Final Error Computation Module (H).

4. Simulation and Synthesis Results

The proposed ASIC has been modelled using Verilog Hardware Description Language. The correctness of the chip was verified by simulating its behavior using Verilog-XL simulator. After obtaining satisfactory results, the logic synthesis has been carried using *Synergy* synthesizer.

We experimented with 64 x 64 images. The original and reconstructed images are shown in Fig. 3 and the PSNR[2], compression ratio (obtained after a post-processing not implemented in hardware) and encoding times are listed in Table 1 . The results obtained from logic synthesis are summarized in Table 2. The *Cell Area* is shown in 10^{-12} sq m. The total area is 4102855.98 X 10^{-12} sq m. This will require a square of side 2.03 mm. The encoding times for the simulation are obtained in number of clock cycles, which is then converted to seconds assuming 50 MHz clock rate.

5. Concluding Remark

The proposed architecture has been designed using *Verilog Hardware Description Language* and has been simulated and synthesized. The entire architecture contains few

[2] PSNR is defined as the ratio of the maximum image power to the difference image power as expressed in dB and the compression ratio is percentage reduction in image size over the original image size.

Table 1. Results in terms of Quality, Speed and Compression

Image	PSNR (db)	Compression (%)	Encoding Time (sec)
aero_decimated	27.14	88.9	0.12
bird_decimated	16.98	86.4	0.32
train_decimated	22.45	87.3	0.26
aero_fragment	32.16	87.6	0.18
bird_fragment	32.12	87.8	0.30
train_fragment	28.56	88.0	0.41

Table 2. Gate Count and Area

Cell Type	Total Count	Total Area (10^{-12} sq.m.)
ADDER	120	1200000.00
SUBSTRACTOR	94	940000.00
OPEN COLLECTOR	1260	402647.68
FLIP FLOP	1123	922181.76
INVERTER	939	146033.28
MULTIPLEXER	399	186157.44
NAND	1006	230437.12
NOR	304	75398.72
TOTAL	5245	4102855.98

major hardware components like some high speed adders, eight array multipliers, few shift registers, multiplexers and comparators. So it can be realized in a single VLSI chip. The proposed chip is meant to run at a clock speed of about 50 MHz.

References

[1] A.Jacquin. Fractal image coding : A review. *Proceedings of the IEEE*, pages 1451–1465, October 1993.

[2] K. Hwang. *Computer Architecture - Principles, Architectures and Design*. John Wiley and Sons.

[3] A. Jacquin. *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding*. PhD Dissertation, Gorgia Institute of Technology, Atlanta, 1989.

[4] L. H. M. Barnsley. *Fractal Image Compression*. A.K. Peters, 1992.

[5] Y.Fisher. *Fractal Image Compression - Theory and Applications*. Springler-Verlag, 1994.

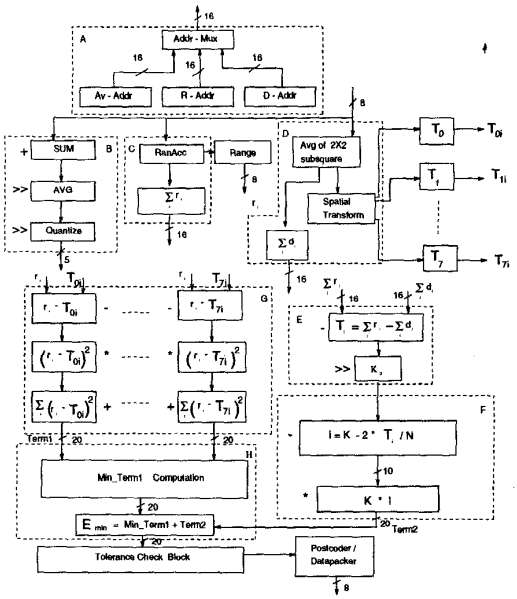

Figure 2. Block Diagram of the Chip

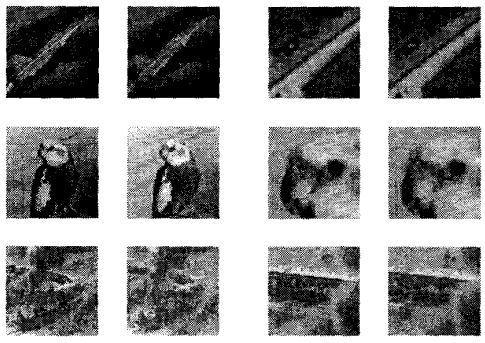

Figure 3. (a) Full Images (b) Image Fragments - Original and Decoded