Design of Fractal Image Compression on SoC

A. Jedidi, B. Rejeb, M. Abid
CES Laboratory, National School of Engineering, University of Sfax, Tunisia
ahmedjedidi@yahoo.fr

Abstract— **The technological revolution during the last years has carried out a great evolution especially in the multimedia domain. Nowadays, almost everyone benefits from various video applications such as TV broadcasting and video conferencing. This progress involves particularly an increase in the capacity of digital data transmission. As a result, data compression became increasingly significant for storage and transmission.**

In this paper, we present an algorithm for fractal image compression based on SOC in real time. The algorithm consists of a hardware and software part. The hardware part supports an expensive calculation; therefore it is conceived on RTL level. The coding algorithm was implemented on the Altera chard STRATIX-I. The functional blocks were implemented with clock rate of 410 MHz with a maximum flow data input of 13.2 Gbit/s.

Index Terms— **Fractal image compression, SOC, FPGA, Real time, RTL level design, Functional simulation, Temporal simulation.**

I. Introduction

THE fast development of the data-processing applications was accompanied by a significant increase in the use of the digital images, in particular in the field of the multimedia, the plays, the transmission satellites or the medical imagery.

Due to their large size, digital images raise many problems for their transmission and storage. For example an image 512x512 pixels (value of the pixel is 8 bits) necessitates 2MB of memory capacity. Also the real time which becomes more and more requires nowadays then the compression of image is necessary.

Currently several methods of image compression are proposed (JPEG, wavelets, etc.) in order to have a good compromise between the compression ratio and the quality of rebuilt image. Worthly mentioning is the **fractal image coding** method which is based on the fractal theory. The basic idea of this method is that the image can be reconstructed using its self similarity features.

it is noted that the phase of integration of the functionalities offered by the multimedia applications in the systems embarked in real time (i.e. the systems which carry out several calculations within a very limited time such as the portable telephones, TV, reader DVD etc) became increasingly complex [1] and [7]. The implementations of the compression methods require the performance in term of computing power, flexibility, cost and time to market.

In this paper, we propose a hardware software design of fractal image compression. First, we briefly present the key concept of the hardware software fractal coding. Then, we mention internal architecture of hardware blocks. After that, we present the results of simulation and implementation of the system in Stratix chard. Finally, we conclude the paper and we present open direction for future work and

II. Fractal coding

The fractal theory applied to the image processing field is based on the iterated function system (IFS) and has been used mainly for data compression [12] and [15]. The basic idea of fractal coding is to exploit the redundancy given by the self-similarities always contained in natural images. The fractal image can be seen as a collage made up of copies of parts of an original image that have been transformed through opportune geometric and massive transformations (that is, luminance or contrast shift). The mathematical foundation of this technique is the general theory of contractive iterated transformations, proposed in [2] and [10].

To understand quantitatively the principal encoding algorithm, consider the flow chard of the fractal image coder as shown in Figure 1. Encoding algorithm is composed of two main parts: software and hardware. The hardware part is called self similarity search [3] and [5].

The software part is made up of four blocks: image partition, domain blocks classification, domain blocks reduction and range blocks classification [8] and [9].

First the image is partitioned in non overlapping parts of size BxB, range blocks, and in overlapping blocks of size 2Bx2B, domain blocks. The partition begins by range blocks of size 16x16 pixels and domain blocks of size 32x32 pixels. Then the domain blocks are classified using the luminance-variance features method, where the blocks are classified in 72 classes. Afterwards the number of domain blocks is reduced with method of Domain Pool Reduction. The main idea is the scanning of the domain pool in order to remove similar or unused domains. The range blocks are classified in 73 classes: those blocks are classified in 2 main classes, shade and non-shade.

For shade blocks, no self-similarity search is needed, only the mean pixels value is used for encoding of the range block. For non-shade blocks, further classification

in 72 classes based on luminance-variance features is performed like as classification of domain blocks.

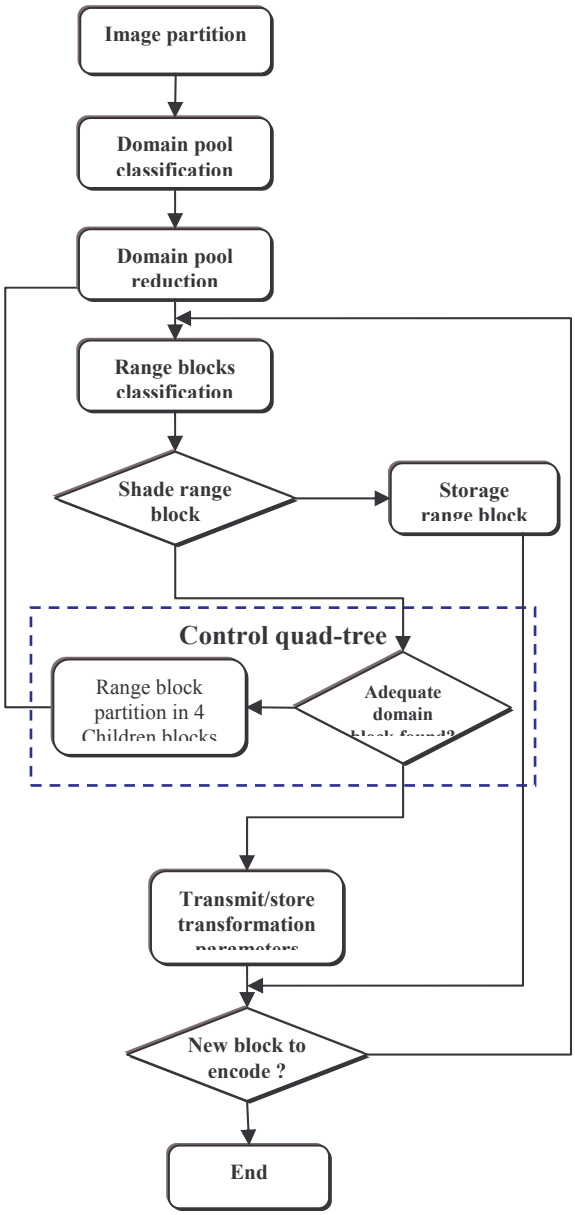

Figure 1: Encoding algorithm flowchart

Finally a process of self-similarity search is begun, if the best domain block for this range is found i.e. the error given by the equation (2) is less a predetermined tolerance, the location of this domain and its transformation parameters (α , β and the symmetry operation defined in the classification for a given R_i and D_i) are stored and the process is repeated for the next range square. However, if the predetermined tolerance was not satisfied, the range square is subdivided into four equal blocks. This control quad-tree process continues until the tolerance condition is satisfied, or range square of the predetermined minimum size 4x4 pixels is reached.

The self-similarity search between range and domain blocks is ensured by the Mean Square Error:

$$MSE = \frac{1}{B^2}\left(\sum_{1 \le i,j \le B} \left(R_{i,j} - \left(\alpha D_{i,j} + \beta \right)^2 \right) \right) \quad (1)$$

Where:

- $R_{i,j}$ and $D_{i,j}$ are the pixels values of the range and domain blocks at coordinates (i,j).
- α and β are the contrast and the offset parameters.
- B is the blocks width and height.

The optimal set of parameters for a given couple of range and domain blocks is found setting the partial derivates according to the contrast parameter α and the brightness β to zero [3].

$$\frac{\partial MSE}{\partial \alpha} = 0 \text{ and } \frac{\partial MSE}{\partial \beta} = 0 \quad (2)$$

Finally the optimal offset and optimal contrast parameters are given by:

$$\alpha_{opt} = \frac{C_{DR}(0,0)}{\sigma_D^2} \quad (3)$$

$$\beta_{opt} = \mu_R - \alpha_{opt} . \mu_D \quad (4)$$

Where:

- μ_R and μ_D are respectively the mean values of the range R and domain D,

$$\mu_R = \frac{1}{B}\sum_{i=1}^{B} r_i \; ; \; \mu_D = \frac{1}{B}\sum_{i=1}^{B} d_i \quad (5)$$

- C_{DR} is the covariance of R and D,

$$C_{DR} = \frac{1}{B}\sum_{i=1}^{B} (r_i - \mu_R)(d_i - \mu_D) \quad (6)$$

- σ_D is the variance of D

$$\sigma_D = \frac{1}{B}\sum_{i=1}^{B} (d_i - \mu_D)^2 \quad (7)$$

The optimal parameters that will be used for encoding of R are given by those of the

$$\min_{D \in P_d}(MSE) = \min_{D \in P_d} \begin{pmatrix} (\Delta\beta + \Delta\alpha . \mu_D)^2 \\ -(\Delta\alpha)^2 . \sigma_D^2 - \alpha_{quan}^2 . \sigma_D^2 \end{pmatrix} \quad (8)$$

Where:

- $\Delta\alpha = (\alpha_{opt} - \alpha_{quan})$ and $\Delta\beta = (\beta_{opt} - \beta_{quan})$, α_{quan} , β_{quan} are α_{opt} and β_{opt} after quantization.
- P_d is the set of domains in one class

The software part is composed by four blocks: image partition, domain blocks classification, domain pool reduction and range blocks classification. This part is realizable in software and offers the real time criterion. On the other hand the hardware part or self-similarity search does not respect the aspect of the real time when

it is implemented in software because it needs a powerful calculation:

The computational complexity of the optimal parameters calculation is of order of o(N), where N is the block size.

The computational requirements, i.e. the number of additions, multiplications and divisions needed for the implementation of the optimal contrast parameter α_{opt} for blocks size N using the equation (3) are amounted to:

- 5N additions
- 3N+1 multiplications
- 2 subtractions
- 7 divisions

According to the equation (4), the number of operations needed for the implementation of the optimal offset parameter β_{opt} using the pre-calculated α_{opt} is amounted to one multiplication and one subtraction.

In the same way the Mean Square Error for a defined couple of range and domain blocks can be calculated using equation (1). The computational requirements are:

- N additions
- 2N multiplications
- N subtractions
- 1 division

The number of MSE to be calculated during the encoding depends on the image size, the blocks size and the search sliding step. For example the encoding of the image Lena 256x256 using 8x8, 16x16 and 32x32 domain blocks and a search step of 4 pixels needs of about 200000 MSE to be calculated. That's why the implementation of the part self-similarity search is necessary in hardware in order to accelerate the time computing[4].

III. TARGET ARCHITECTURE

A mono chip system, still called SoC "System-one-Chip" can be defined as a complex integrated circuit which integrates complexes and heterogeneous component on the same silicon part. With the difference of the traditional systems the mono chip systems are dedicated to specific applications and cut to measure to satisfy only the needs for the application. Compared to a system on ordinary chard, SOC employs only one chip reducing the total cost of encapsulation which accounts for approximately 50% of the total cost of manufacture. These characteristics as well as weak consumption and the short duration of design allow a fast setting on the market of an economic and powerful product "time to market" [11] and [13].

Figure 2 presents the basic architecture of our chip. It is composed of: processor, memory and hardware dedicate blocks. The units communicate between them by means of a bus. First, the processor provides the data with the various hardware blocks and recovers the results to safeguard them in the memory. That's why the processor is only connected with the memory. Initially, it takes the untreated data.

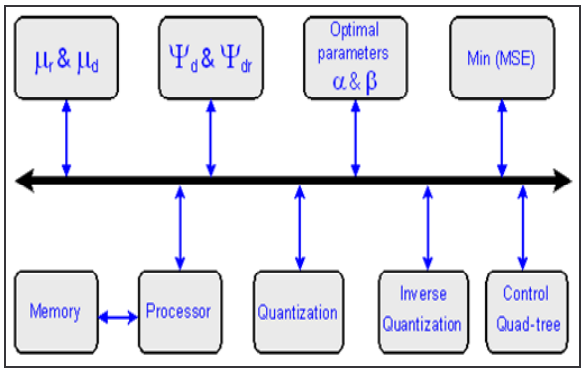

Figure 2: Architecture of SOC

Then, it slings defined classification before giving them to the various hardware blocks. Finally, it recovers the results in order to safeguard them in the memory.

The hardware blocks are composed of seven elements:

1- Block μr &μd: is calculating the mean values domain and range.
2- Block Ψd &Ψdr: is to calculate the mean of product respectively domain/domain and domain/range
3- Optimal parameters α & β: its objective is to calculate the optimal parameters
4- Quantization and Inverse quantization: the first block is quantized the optimal parameters while the second block is un-quantized.
5- Block min(MSE): this block calculate the minimum of mean square error
6- Control quad-tree: this block is responsible for controlling the system of quad-tree.

IV. DESIGN OF THE SELF-SIMILARITY SEARCH:

As already mentioned, self-similarity search is implemented in hardware dedicates blocks. One of the serious problems is the synchronisation of the data flow between the different hardware blocks. In order to satisfy this requirement, we created a global State machine. Let's start by developing the different hardware dedicates blocks.

A. Hardware dedicates blocks architecture:

The proposed architecture for self-similarity search is shown in figure 3. It is composed by four stages implemented by serial-parallel architecture: average value, variance & covariance, the optimal parameters and validation stage.

i. Stage of average value: it is composed by four blocks. Each block is responsible to calculate: the mean value of domain, mean value of range, mean product of domain and mean product of domain & range. These values are respectively denoted by: μ_r , μ_d , ψ_d and ψ_{dr} .

μ_r , μ_d , ψ_d and ψ_{dr} are calculated according to the size of the image blocks to be coded (16x16, 8x8 and 4x4 pixels). The inputs signals are the various pixels of domain and range block and start_M.

ii. Stage of variance & covariance: this stage contains two modules. First, the variance it's role calculates the square of variance of domain. That the second, the covariance is calculating the covariance of range and domain. These different blocks are controlled by start_COV and start_VAR.

iii. Stage of optimal parameters: it is composed of two modules optimal contrast parameter and optimal offset parameter. The square of variance of domain and the covariance of domain & range are the inputs signals of the first module, whereas the output is the optimal contrast parameter (α_{opt}). Optimal offset parameter module has these inputs signals: average value of domain, average value of range after a shift of four clock cycles, clock and optimal contrast parameter. This module is to calculating optimal offset parameter (β_{opt}).

iv. Stage of validation: this stage is responsible to validate the different optimal parameters by using minimum (MSE). Figure 4 presents the internal architecture of this block. During the encoding process, only the optimal parameters and the domain position should be sent to the encoder. For storage and transmission, the optimal parameters should be quantized in order to increase the compression ratio. The uses of the quantized parameters at the decoder make it necessary to correct the equation (1) by a quantization error. That's why optimal parameters should be quantized and versed quantized in order to determinate the error of MSE.

Finally, the outputs of this block are the optimal parameters (α_{opt} and β_{opt}), minimum of MSE and the identification of coded block. These signals are directly connected to the processor.

There are also two blocks of shift (block_shift_1 and block_shift_2). These blocks have the role to synchronise the various signals to have a correct process. Indeed, certain blocks need the signals which come in various stages: for example the block optimal offset parameter needs three signals: two signals come from the first stages: µr_pixels and µd_pixels. On whereas, the third signal comes from the third stage α_param_opt; therefore it needs to shift the first two signals in order to have correct result, figure3.

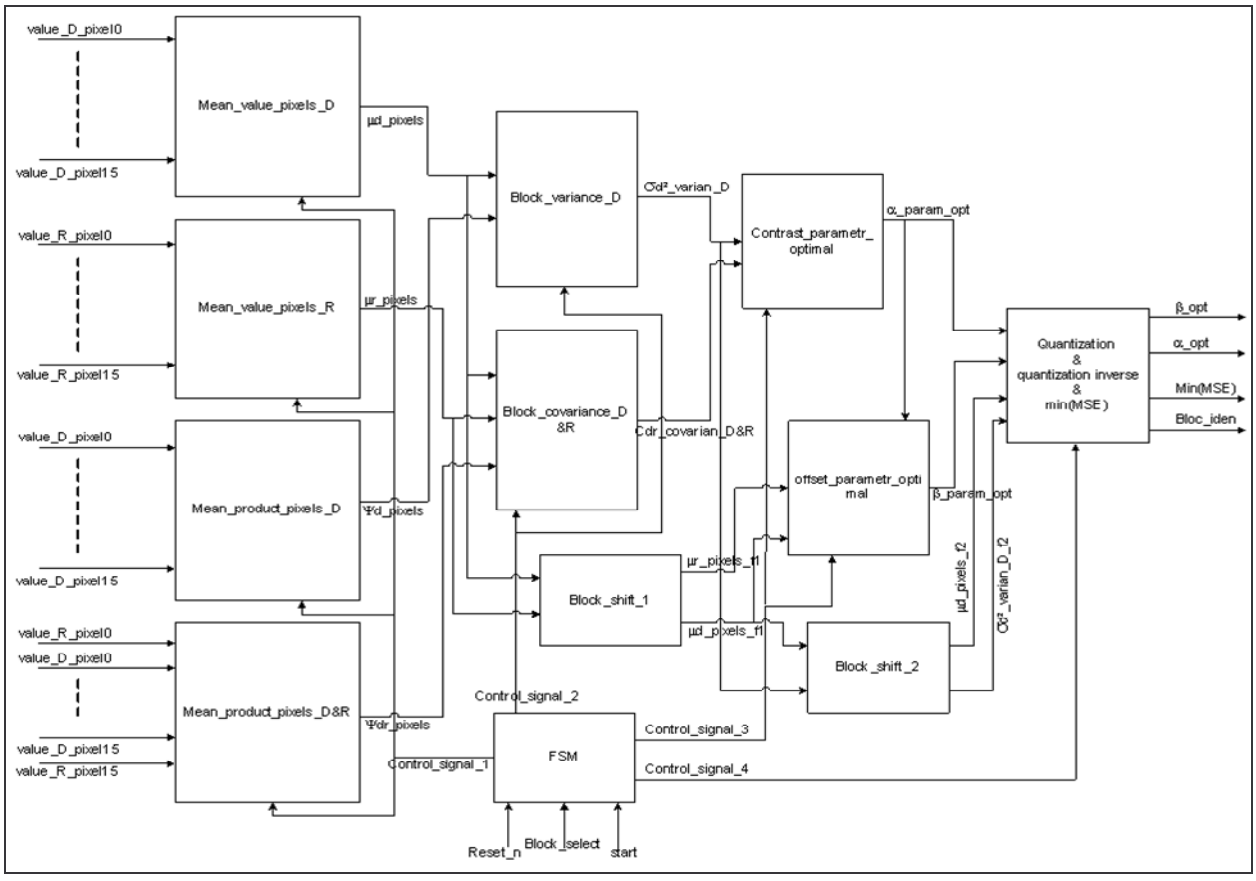

Figure 3: design of self-similarity search

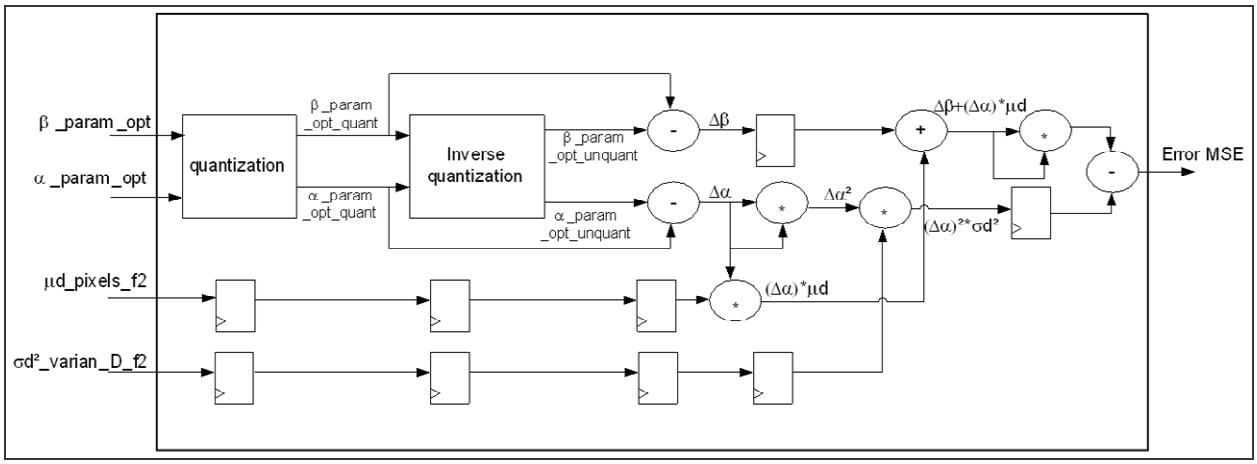

Figure 4: quantization, un-quantization and min (MSE) design

B. *Global State Machine:*

The global architecture is controlled by different states machines. These states machines have a very significant role in this module self-similarity search.

For simplicity of the design of the state machine, it is important to divide it into two modules: Global State Machine and six sub states machines: fsm_M, fsm_COV, fsm_VAR fsm_Contarst, fsm_Offset and fsm_MSE, comes respectively in these blocks: average value, variance, covariance, optimal contrast parameter, optimal offset parameter and validation stage.

A block diagram of a global state machine is shown in figure 5. It is responsible to generate the different starts signals of six sub states.

Now considering the situation when the block_select is set to "01" (image block size 8x8). Figure 6, present the simulation of this situation. Initially, start_M passes to 1 when counter is equal to x "1". This signal activates the state machine of the various modules of first stage (average value). When the counter is equal to x"7", start_Var and start_Cov passes to 1 and activates respectively the module variance domain and covariance range& domain. After three cycles of clock start_contrast is activated. Therefore the block of optimal contrast is beginning to work. The block of optimal offset is started the work when start_offset receive 1. Finally, when the counter is equal to x "0d"; start_Min(MSE) passes to 1 then module of validation begins the work and after two cycles we have the various optimal parameters to outputs.

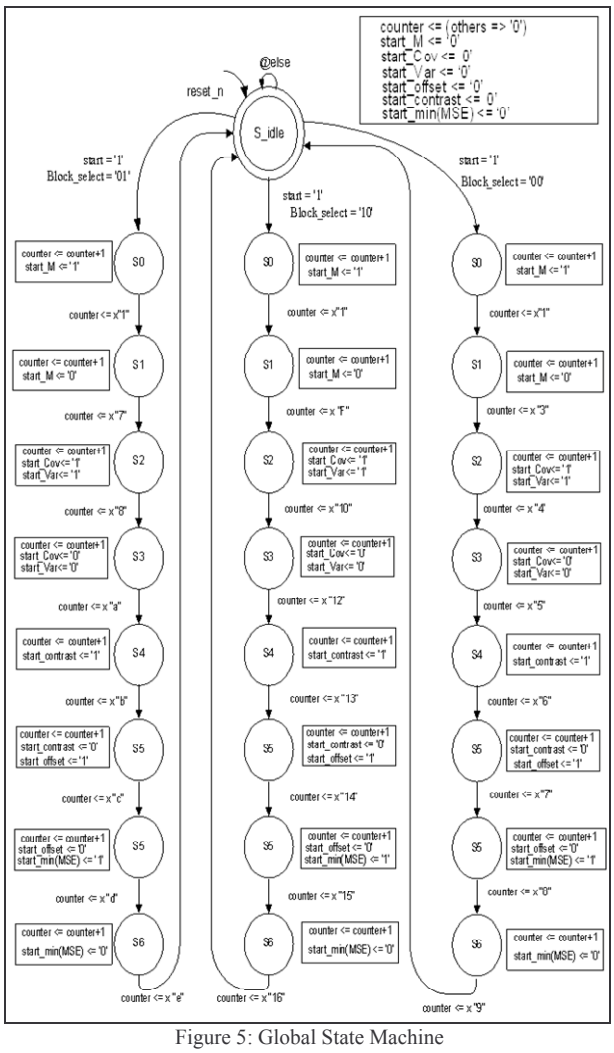

Figure 5: Global State Machine

V. IMPLEMENTATION RESULTS

To implement the two large modules: software and hardware (self-similarity search) we use chard FPGA STRATIX of the Altera firm which uses processor NIOS.

The Stratix family of FPGAs is based on a 1.5-V, 0.13-µm, all-layer copper SRAM process, with densities of up to 79,040 logic elements (LEs) and up to 7.5 Mbits of RAM. Stratix devices offer up to 22 digital signal processing (DSP) blocks with up to 176 (9-bit × 9-bit) embedded multipliers, optimized for DSP applications that enable efficient implementation of high-

performance filters and multipliers. Stratix devices support various I/O standards and also offer a complete clock management solution with its hierarchical clock structure with up to 420-MHz performance and up to 12 phase-locked loops (PLLs).

We use also environment Nios-II and Quartus for creation, compilation, simulation, and prototyping on this FPGA card [14].

For architecture self-similarity search we propose several designs on RTL level in order to have a synthesisable architecture which offers a good compromise between the frequency, the area, the time execution and the data flow. The presented architecture has been simulated using Active VHDL. Thus, we make a functional simulation to validate the correct operation of this architecture. Then we pass to the temporal simulation which makes it possible to guarantee an implementation success of 95% on the STRATIX FPGA cart.

The total implementation in FPGA for the fractal image coding on the STRATIX chard offers these results:

- Total ALUTs: 7124
- Total register: 4559
- Total pins: 177
- Total memory bits: 571136

Table 1 shows the area, the latency time in clock cycle and the throughput for self-similarity search implementation. The proposed architecture has a frequency of 410MHz with a data flow is 13.2 Gbit/s. therefore this architecture can be used in real time video compression, because 25 frames/s needed 13 Gbit/s but we use the reduction of data by 40% proposed by [3].

VI. CONCLUSION:

The main objective of the presented work was real-time implementation of the fractal image compression on SOC. That's way we divided the encoding algorithm into two parts: software and hardware (self-similarity search).

We conceived the self-similarity search architecture, and then we made functional and temporal simulation. Afterwards we implemented the algorithm of coding on the STRATIX chard. In experiments we had a frequency of 410MHz with a data flow of 13.2 Gbit/s.

We obtained the real time coding of video image with exploiting only the ¼ totality of the resources of the STRATIX chard.

There are several open issues for future research. One is how to integrate the algorithm for estimate movement in the encoding algorithm of fractal compression, and how to improve the algorithm of fractal compression to adapt in HDTV.

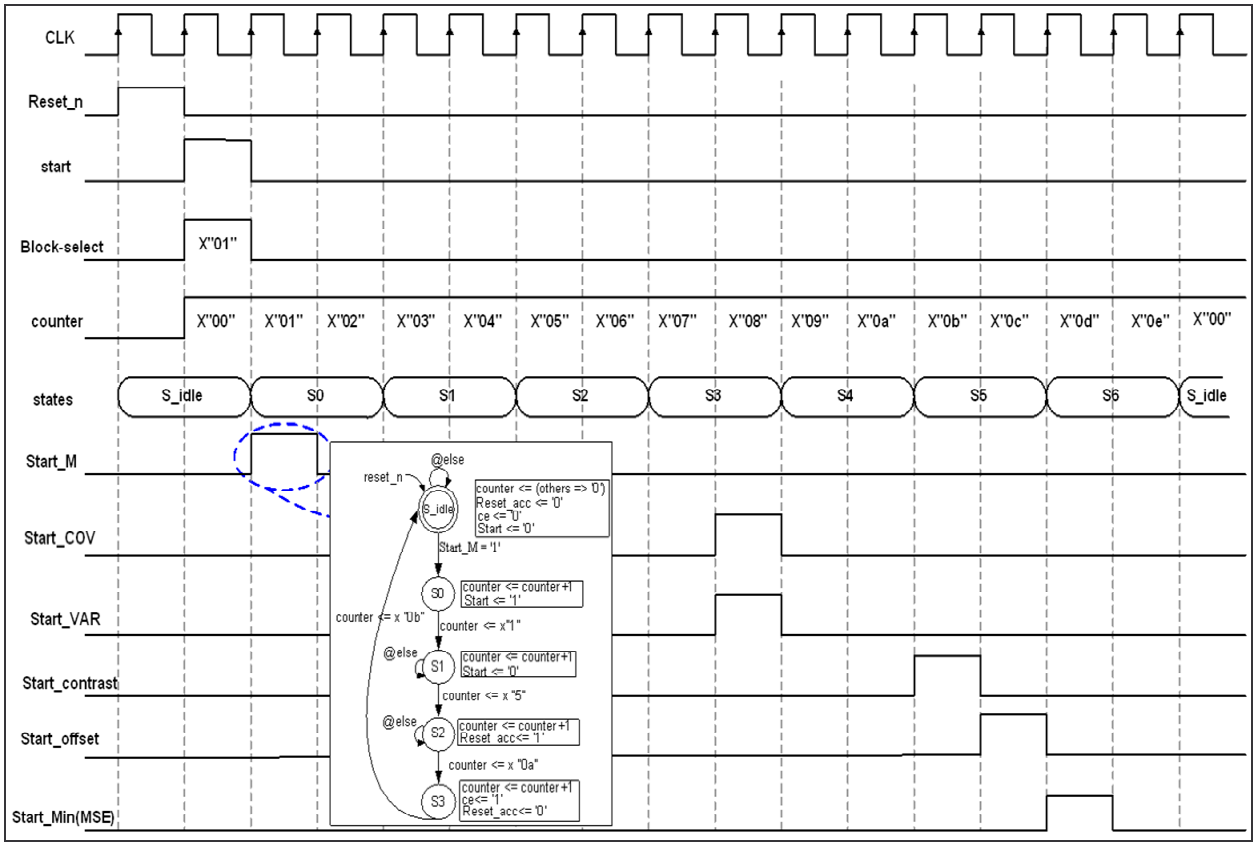

Figure 6: simulation diagram for Global State Machine when block_select is equal to x"01"

REFERENCES:

[1] A. Martínez Ramírez, A. Díaz Sánchez, M. Linares Aranda, and J. Vega Pineda: "An Architecture for Fractal Image Compression Using Quad_tree Multiresolution", ISACS, IEEE 2004.

[2] M. Barnsly: "Fractal modelling of real word image", Spring-Velag, New York, 1998.

[3] B. Rejeb, W. Anheir : "A New approach for the Speed up of Fractal Image Coding", IEEE DSP conference, July 1997, Santorini, Greece.

[4] Saraju P. Mohanty1, Renuka Kumara C., and Sridhara Nayak : "FPGA Based Implementation of an Invisible-Robust Image Watermarking Encoder", Springer-Verlag Berlin Heidelberg 2004

[5] B. Rejeb, H. Henkelmann and W. Anheier: "Real-Time Implementation of Fractal Image Encoder using Residue Number System", IEEE NORSIG'200, Jun 2000, Sweden.

[6] Mohsen Ghazel, George H. Freeman, and Edward R. Vrscay : "Fractal-Wavelet Image Denoising Revisited IEEE Transactions on image processing, VOL. 15, NO. 9, SEPTEMBER 2006.

[7] S.Mozaffari, K.Faez, M.Ziaratban: "Character Representation and Recognition Using quadtree-based Fractal Encoding Scheme", Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition (ICDAR'05).

[8] M. Ghazel, R.K. Ward, R. Abugharbeih, E. R. Vrscay, G. Freeman: "Simultaneous Fractal Image Denoising and Interpolation", IEEE 2005

[9] Geoffrey M. Davis, Member, IEEE: "A Wavelet-Based Analysis of Fractal Image Compression", IEEE Transactions on image processing, VOL. 7, NO. 2, FEBRUARY 1998

[10] In Kwon Kim, Member, IEEE, and Rae-Hong Park, Member, IEEE: " Still Image Coding Based on Vector Quantization and Fractal Approximation", IEEE Transactions on image processing, VOL. 5, NO. 4, APRIL 1996

[11] Wonjong Kim, June-young Chang and Hanin Cho : "piplined scheduling of functional HW/SW modules for platform-based SOC design". ETRI journal, volume 27, Octobre 2005

[12] Henry, Xiao : "Fractal Compression", Related Topic Report Queen's University Kingston, Ontario, Canada April 2004

[13] Gharssali F. "Conception des interfaces logiciel-matériel pour l'intégration des mémoires globales dans les systèmes monopuces ". Thèse de doctorat, GRENOBLE, Juillet 2003.

[14] Xilinix, Virtex-II Pro™ Platform FPGAs, Complete Data Sheet. DS083, Product Specification, April 22, 2004.

[15] A.E Jaquin, "Image coding based on a fractal theory of iterated contractive image transformation", IEEE Trans. On Image Processing, vol. 1, No 1, January 1992.