LAR method: from algorithm to synthesis for an embedded low complexity image coder

Olivier Déforges, Marie Babel

LAR method : from algorithm to synthesis for an embedded low complexity image coder

Olivier Déforges, Marie Babel
CNRS UMR 6164 IETR/INSA Rennes, 20 Av des Buttes de Cöesmes
35 043 Rennes Cedex, France
Email : odeforge@insa-rennes.fr

***Abstract*— Embedded image codecs have generally strong constraints in terms of speed, latency and complexity. We introduce in this paper a dedicated coder implementation of the LAR coding technique, suitable for high image compression rates. The general concepts of the LAR codec is firstly described. Then, a low complexity architecture is detailed. This architecture is composed of parallel and pipelined stages, and presents limited requirements for both memory and computation power. Comparative results with FPGA solutions for JPEG and JPEG2000 are finally discussed.**

I. Introduction

Despite many drawbacks and limitations, JPEG is still the most commonly-used compression format in the world. JPEG2000 overcomes this old technique, particularly at low bit rates, but at the expense of a significant complexity overhead. Therefore, the JPEG normalization group has recently proposed a call for proposals on JPEG-AIC (Advanced Image Coding) in order to look for new solutions for still image coding techniques [1]. Its requirements reflect the earlier ideas of Amir Said [2] for a good image coder : compression efficiency, scalability, good quality at low bit rates, flexibility and adaptability, rate and quality control, algorithm unicity (with/without losses), reduced complexity, error robustness (for instance in a wireless transmission context) and region of interest decoding at decoder level. Among them, the reduced complexity is maybe the most important criteria for embedded systems such as camera or mobile phone. The complexity of a coding system does not only impact on the processing time, which is a critical point for digital camera, but also on the consumption of the embedded system and therefore on its self-sufficiency time.

To reduce both consumption and execution time, some dedicated circuits exist for JPEG and JPEG2000 coders, most of the time relying on ASIC technology. However, these solutions induce two major drawbacks :

– no system evolution,
– high design cost.

An alternative can be the use of reprogrammable components such as FPGAs, but requirements significantly differ between JPEG and JPEG2000. JPEG is characterized by a low complexity, and relating IP blocks exist which can be integrated in low capacity FPGAs (for instance an ALTERA FLEX 10K) [3]. JPEG2000 requirements both in terms of memory and computation power are much more important : only large FPGAs, and therefore expensive and power consuming, enable its integration [4].

In [5], we proposed an original scheme call LAR (Locally Adaptive Resolution) able to perform efficient lossy compression, enabling an unusual hierarchical region representation (without any shape description). Then, in [6], we described an extension of a more efficient scalable multi-resolution solution in terms of both lossy and lossless compression. This paper more focuses on the implementation aspect of the method. In particular, we detail an FPGA implementation of part of the LAR coder, well suited for high ratio compression.

This paper is organized as follows. Section 2 gives an overview of the LAR coding method and more particularly on the FLAT LAR. Efficient dedicated hardware solution and synthesis results are presented in section 3, and compared to FPGA solutions designed for JPEG and JPEG2000. Finally, section 4 concludes the paper.

II. The LAR coder

A. General concepts

The LAR method relies on the idea that the resolution can be locally adapted to activity : when local luminance remains uniform, the resolution can be low, whereas for high activity areas, good image representation requires a finer resolution. The second principle in the LAR method relies on the fact that an image can be seen as the superposition of two complementary components :

$$I = \bar{I} + (\underbrace{I - \bar{I}}_{E}) \tag{1}$$

$\bar{I}$ represents the image global information (local average value for instance), estimated on a given support, and E is the local variation around it (say local texture). The range of E is dependant on two main factors :

1) the image local activity,
2) the support dimension of $\bar{I}$.

Moreover, if we assume that an image can be roughly seen as the composition of homogeneous areas and contours, then the sole adaptation of the support leads to low dynamic range of E in uniforms areas. On the other hand, E dynamic range will be wide on contours as soon as the support of $\bar{I}$ is superior to one pixel.

In connection with the previous remarks, the main concepts of the LAR method relies on a two layers codec : the first layer called FLAT LAR represents the global information, whereas the second stage, called error spectral coder, provides the local texture. By construction, the LAR method enables two layers of scalability. Figure 1 shows the associated global scheme.

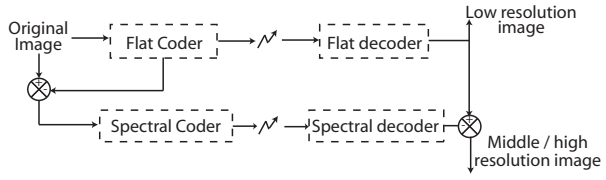

Fig. 1. Global LAR scheme : codecs FLAT + spectral layers

This work is mainly focuses on the synthesis of the first layer, which will be detailed in the following.

B. The flat coder

The expression "FLAR" means that the representation of $\bar{I}$ in first layer is performed by local flat approximations. As the objective of this FLAT codec is only to compress the global image information, it clearly addresses high compression rates. The relative image representation aims to distinguish between contours and the remaining of the image, and to adapt the support of $\bar{I}$ such as the rebuilt image remains visually acceptable and presents reduced error E especially in uniforms areas. The support takes here the shape of squared blocks.

The global scheme of the FLAT coder is given on figure 2. It relies on a variable block size representation of the image, in which blocks are filled by their mean graylevel value. If the technique can seems to be well-known, our approach presents particularities that are detailed in the following. All data produced by the different processing steps (partitioning, prediction errors, ...) are further losslessly compressed by a low complexity entropy coder, which will be later presented.

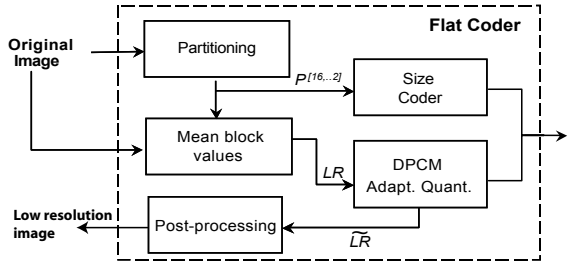

Fig. 2. FLAT coder scheme

1) Partitioning: Every systems relying on a variable block size representation of images induce an activity criterion (or homogeneousness) and a particular partition topology. In the following, we consider the Quadtree partition $P^{[N_{max}\dots N_{min}]}$, in which N_{max} and N_{min} respectively represent the maximal and minimal allowed block sizes, expressed as power of 2. $I(x,y)$ denotes a pixel in the image with coordinates (x,y), and $I(b^N(i,j))$ is the block $b^N(i,j)$ in I such as :

$$b^N(i,j) = \{(x,y) \in N_x \times N_y | \quad N \times i \leq x < N \times (i+1), \\ N \times j \leq y < N \times (j+1)\}. \tag{2}$$

Among all existing coding techniques, some of them also consider an image partitioning. For instance, the intra mode of MPEG4-AVC enables two sizes for blocks, *i.e.* 4 and 16. It can be also seen as a $P^{[16,4]}$ Quadtree partition. The block size in AVC is selected according to distortion/rate criteria, from a PSNR point of view [7]. Other methods involve a finer partitioning such as tree decomposition solutions : the decomposition starts at the highest level of the tree (maximal size), and a node is decomposed into four sons since the activity threshold is exceeded. Several homogeneousness criteria have been proposed [8], [9], but in most of cases they rely on the estimation of L_1 or L_2 norms distance between a given block and its sons.

In our approach, we have adopted a different criterion, as the notion of activity is closely here associated to the presence of contours or not. Thus, the activity estimation is performed by a morphological gradient (difference between the maximal and minimal values inside a block).

We consider the Quadtree partition $P^{[N_{max}\dots N_{min}]}$. Let $min[I(b^N(i,j))]$ and $max[I(b^N(i,j))]$ be respectively the minimal and maximal values in block $I(b^N(i,j))$.

The sizes image in all points is given by

$$Siz(x,y) = \\ \begin{cases} N \in [N_{max} \dots N_{min}[\\ \quad \text{if } |max[I(b^N(\lfloor \frac{x}{N} \rfloor, \lfloor \frac{y}{N} \rfloor))] - min[I(b^N(\lfloor \frac{x}{N} \rfloor, \lfloor \frac{y}{N} \rfloor))]| \leq Th \\ \quad \text{and if } \exists (k,m) \in \{0,1\}^2 \ / \\ \quad\quad |max[I(b^N(\lfloor \frac{x+k}{N/2} \rfloor, \lfloor \frac{y+m}{N/2} \rfloor))] - min[I(b^N(\lfloor \frac{x+k}{N/2} \rfloor, \lfloor \frac{y+m}{N/2} \rfloor))]| \\ \quad\quad > Th \\ N_{min} \text{ otherwise,} \end{cases} \tag{3}$$

with Th denoting the activity threshold.

The sizes image directly provides a coarse image segmentation map : blocks of size N_{min} are mainly located upon edges and highly textures areas of the image. We will show in the following that this feature enables an adapted quantization in the coding process.

2) Block mean value: A low resolution image LR is reconstructed by representing each block by its mean value. Consequently, for each pixel $p(x,y)$ we get :

$$LR(x,y) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{m=0}^{N-1} I(\lfloor \frac{x}{N} \rfloor \times N + k, \lfloor \frac{y}{N} \rfloor \times N + m), \\ \text{with } N = Siz(x,y) \tag{4}$$

3) Block mean value encoding by DPCM approach:

a) Block mean value quantization: Compression techniques relying on distortion/rate optimization try to find the best compromise between the coding cost and global errors, but only from a $PSNR$ or MSE point of view, without any consideration about the human vision. Nevertheless, experi-

mentations have demonstrated that this human eye is much less sensitive to luminance and chrominance variations in areas such as edges (high visual frequencies) than in uniform areas (low visual frequencies) [10], [11]). This principle is simply used in our coding scheme by performing a quantization adapted to the block size. Let q_N be the quantization step for blocks of size N. Then using a set of values, such as $q_N = \frac{q_{N/2}}{2}$, leads to a similar visual quality in the image.

b) Block graylevel mean value prediction: The block mean value encoding is directly realized in the spatial domain, by a DPCM (Differential Pulse Coding Modulation) approach. This choice has been firstly motivated by the simplicity of the coding process, requiring only one regular image scan. Secondly, the block representation provides an interesting a priori about activity areas, which can be useful to adapt prediction.

We have implemented a simple Graham predictor [12] adapted to our context : a linear prediction is performed in homogeneous areas, and a non linear one near contours. The prediction is driven by the local gradient and enables to optimize the predictor according to the context. The predictor is given by :

$$
\begin{array}{l}
\text{For all vertex pixel } p(x,y) \text{ of block } b^N(x,y), \text{ we define the predicted value} \\
\text{of block } \breve{LR}(x,y) \\
\text{from already reconstructed values} \\
\tilde{LR}(x-k,y-m),(k,m) \in \{0,1\} \\
\breve{LR}(x,y) = \\
\left\{ \begin{array}{ll}
\tilde{LR}(x-1,y) & \text{if } |\tilde{LR}(x-1,y-1) - \tilde{LR}(x,y-1)| \\
& \quad < |\tilde{LR}(x-1,y-1) - \tilde{LR}(x-1,y)| \\
& \text{et if } A_N < |\tilde{LR}(x-1,y-1) - \tilde{LR}(x-1,y)| \\
\tilde{LR}(x,y-1) & \text{if } |\tilde{LR}(x-1,y-1) - \tilde{LR}(x-1,y)| \\
& \quad < |\tilde{LR}(x-1,y-1) - \tilde{LR}(x,y-1)| \\
& \text{and if } A_N < |\tilde{LR}(x-1,y-1) - \tilde{LR}(x,y-1)| \\
(\tilde{LR}(x-1,y) + \tilde{LR}(x,y-1))/2 & \text{otherwise.}
\end{array} \right. \\
A_N \text{ is an increasing parameter of } N \\
\text{with : } A_1 = 0; A_2 = 10; A_4 = 20; A_8 = 40; A_{16} = 80;
\end{array}
\tag{5}
$$

Prediction errors quantization process follows the principle mentioned before, with a quantization adapted to the block size. Let $E_{LR}(x,y)$ br the prediction error, $\hat{E}_{LR}(x,y)$ and $\tilde{E}_{LR}(x,y)$ respectively the quantized and dequantized values. Then the associated relationships are defined by :

$$
\left| \begin{array}{l}
E_{LR}(x,y) = LR(x,y) - \breve{LR}(x,y) \\
\hat{E}_{LR}(x,y) = Q\left(E_{LR}(x,y)\right) = round\left[\frac{E_{LR}(x,y)}{q_N}\right] \\
\tilde{E}_{LR}(x,y) = Q^{-1}\left(\hat{E}_{LR}(x,y)\right) = q_N.\hat{E}_{LR}(x,y) \\
\tilde{LR}(x,y) = \breve{LR}(x,y) + \tilde{E}_{LR}(x,y)
\end{array} \right.
\tag{6}
$$

The quantization steps q_N given in table I correspond to possible usable values without major visible degradations on the rebuild image.

Actually, to avoid an accumulation of quantization noise during block mean value normalization (4) and quantization (6), both are performed simultaneously. As values can be expressed as power of 2, the normalization+quantization processes can been implemented as a simple shifting operator. Then, the block mean value estimation step has to provides the sum of pixel values inside the block.

Taille	16×16	8×8	4×4	2×2	1×1
q_N	2	4	8	16	32

TABLE I
QUANTIZATION STEPS ACCORDING TO THE BLOCK SIZE

4) Postprocessing: By construction, the rebuilt LR images present "blocks effects". Nevertheless they significantly differ from perceptible artifacts induced by common coding methods (JPEG, MPEG2 or MPEG4) at low bit rates. Techniques relying on an uniform block decomposition generate high distortions upon contours, which are hardly removable. In our approach, blocks effects appear in uniform areas as a flat information representation, and upon contours as a lack of resolution since $N_{min} > 1$. We have designed a software postprocessing solution able to smooth homogeneous areas while maintaining sharp edges.

Postprocessing is not necessary embedded in the coder. It can be performed off-line or on-line during the decoding process only. It means that the proposed coder is well adapted to applications with hard ressources requirements.

Comparative results are given figure 3 for low bit rates compression. IRCCYN laboratory (Nantes, FRance) has run subjective quality tests between JPEG, JPEG2000 and LAR. Their conclusions reveal that LAR outperforms JPEG at all bit rates, and also JPEG2000 for advanced LAR modes involving chromatic images encoding at a region-level representation **??**.

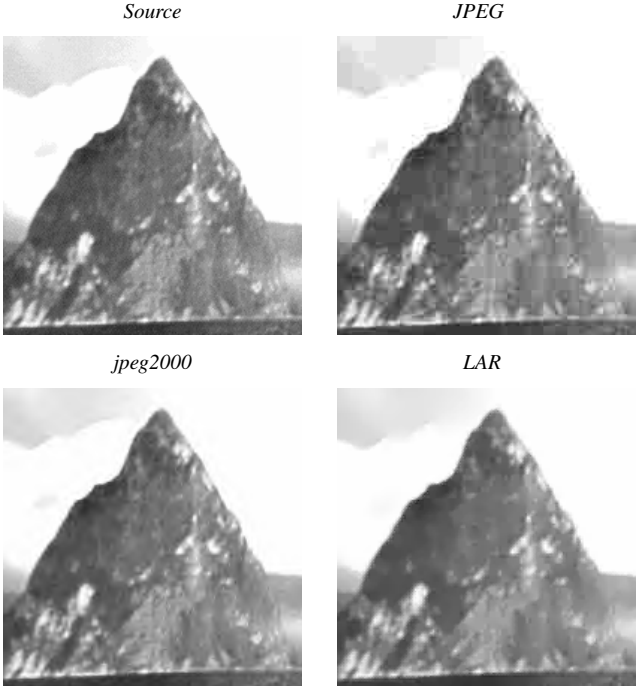

Fig. 3. Comparative results, compression rate =30

5) Entropy coding: In coding systems, symbols are finally losslessly encoded by entropy coders. The principle of these coders consists of using variable length codes according to symbols probability. Entropy coding has an important impact on the performance of the global coding scheme. Many techniques exist, most efficient ones rely on arithmetic coding [13]. An advanced arithmetic coding solution has been integrated in JPEG2000 which actually achieves high compression performances. However, it is also the main limitation for the design of low complexity JPEG2000 codecs.

To decrease global symbols entropy, one has generally to separate symbols with different probability laws. The use of quadtree decomposition enables the approach to act as an "objective context modeling", separating the error prediction laws characterized by high entropy for small blocks, and low entropy for large ones.

We have chosen a Golomb-Rice entropy coding for the low complexity version of the LAR coder. The first reason is that it requires limited computation. The second reason is that Golomb-Rice encoding enables a minimal cost initialization, and therefore is well adapted to small streams of symbols.

Golomb codes G_m with parameter $m \geq 0$ encode an integer number $n \geq 0$ in two parts such as[14] :

– the lowest part $n \, mod \, m$ as a direct binary format,
– the highest part $\lfloor n/m \rfloor$ as an unitary code format.

Rice [15] has presented the particular case when $m = 2^k$, for which coding and decoding operations become very simple. Indeed, on one hand the operation $in \, mod \, m$ consists only of keeping the k lowest bits of n, and on the other hand, the integer division $\lfloor n/2^k \rfloor$ can be performed by k right shifts of n. The code is then composed of series of 1 with a length of $\lfloor n/2^k \rfloor$, a 0 as separator, and the lowest k bits. Thus codes present a global length of $\lfloor n/2^k \rfloor + 1 + k$.

The key factor for Golomb-Rice codes efficiency relies in the optimal choice of the k parameter, which has to be transmitted to the decoder. For symbols initially encoded with a maximum of 8 bits, as many possible values exist for k. The very low computation complexity of the method enables to test all configurations, and to retain at the end the best k value. In practice, a training involving only the 200 first values is enough to select appropriate k value. As Golomb-Rice codes can handle only positive values, sign bits of prediction errors are separately encoded.

Applying Golomb-Rice codes technique to prediction errors streams introduced a cost overhead of about 5% comparing to entropy values. This result remains satisfactory considering the approach simplicity.

III. LAR CODER IMPLEMENTATION

A. DSP implementation

The LAR coder has been firstly implemented into a DSP. Part of research carried out in the laboratory concerns the design of fast prototyping methodologies, under SynDEx development tool [16]. This application is described as a data flow graph and the target architecture is also modelized as a graph. Then the tool can perform the best partitioning and allocation of the application onto the architecture. Without any advanced code and memory optimization, the FLAT LAR coder has then been implemented in a TI TMSC6416 DSP, running at 400 MHz and with 1 Mo internal memory. In this way, a CIF image (352×288 pixels) can be processed in 10 ms (4 times real-time), with a memory use of 800 Ko for both code and data, and a latency time of 9 ms.

B. FPGA implementation

1) Specifications: The objective was to design a dedicated architecture for the LAR coder with limited latency and memory requirements. We imposed a synchronous system with a data flow rate of one pixel incoming and outcoming per each clock cycle. Block sizes has been set ($N_{max} = 8$ and $N_{min} = 2$), as well as quantization values. Image size (X_size, Y_size) and gradient threshold Th are defined as parameters. The integration of the Golomb-Rice coding is currently in progress.

2) General concepts: The first design has consisted of duplicating the initial data flow graph description of the application, rewriting elementary functions into VHDL, and finally performs the synthesis. The major drawback of this approach is that it implies the use of several large buffers to store intermediate images, and increases latency.

A more suitable approach suggests to take advantage of a same image scan for each function (line by line), by performing processing at block level and by pipelining the different stages. The computation process operates on slices of N_{max} lines height, and can be decomposed into two main stages.

– *One values estimation stage* (*min, max*, and *sum*) for blocks of size $N_{max} \times N_{max}$, which induces a fix delay of $(N_{max} - 1) \times X\_size + N_{max}$ for outcoming data, as illustrates the figure 4.
– *One values rebuilding stage* (*DPCM*) for variable size blocks, which can directly follow on the previous stage.

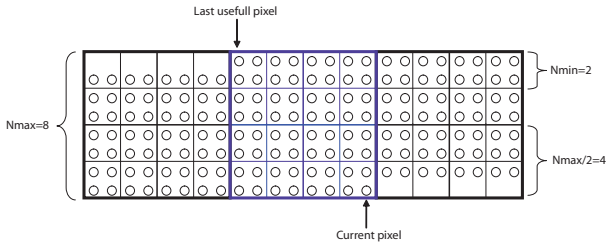

Fig. 4. Blocks processing by raster scans

3) Stages implementation: The global architecture of the system has been redesigned in order to enable a pipeline processing of slices (see figure 5). Between the two mentioned before stages, another one has been integrated, in order to perform current data selection according to the block size. Main stages are pipelined or parallelized in order to increase temporal performances.

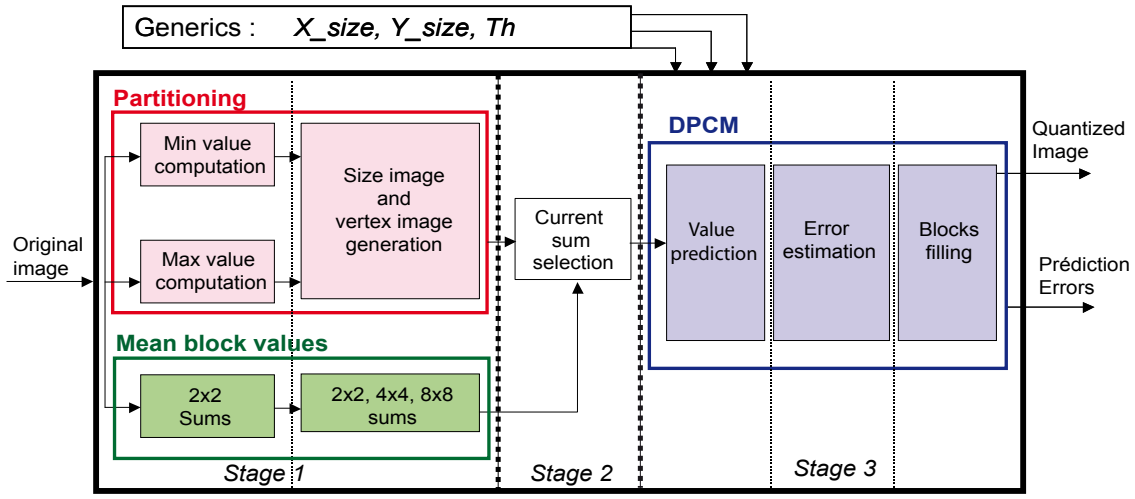

Fig. 5. Proposed parallel and pipeline architecture

a) Stage 1: The estimation of the minimal, maximal values and the sum, for the different possible sizes of blocks, is realized in a recursive and parallel manner. Minimal and maximal values are used only to estimate the local activity by comparing local gradient to the threshold Th. Assuming that all block sizes are initialized to 2×2, only gradients have to be estimated from 4×4 block size according to (3). Thus, for each 8×8 block, it is necessary to store intermediate minimal and maximal values for the four 4×4 blocks inside the block and for the 8×8 block itself (5 values to store). The computation of the sum of blocks follows the same principle, but has also to consider the sixteen 2×2 blocks inside the 8×8 block, which leads to $5 + 16$ values to store. To sum up, the total number of required stored data is equal 26 per block, which has to be multiplied by the number of blocks in an image slice $(X\_size/8)$.

The size image results from the morphological computation and contains for each pixel the selected block size. The vertex image is generated to indicate to latter stages the beginning of a block. A vertex is located at the upper left corner of a block.

b) Stage 2: It is simply a multiplexing stage selecting sum value among the 3 possible ones according to the local block size.

c) Stage 3: The objective of this stage is to perform the DPCM coding. Quantization is adapted to the block size (values are expressed as power of two and stored in buffers). The coding is actually realized for the first pixel of the block (vertex), and for the remaining positions, the vertex quantized value is simply diffused. The stage generates two outputs : the low resolution block image $\tilde{LR}$ and the prediction errors $\hat{E}_{LR}$. Then, the DPCM stage itself is divided into three successive pipelined elementary steps : value prediction, error estimation and quantization, and filling of the block.

C. Implementation results

The proposed architecture has been described in VHDL, and synthesized into a medium FPGA : a xcv600 (Xilinx) including 15000 equivalent gates, and 12 Ko of embedded block memory. The development process was relying on *Modelsim* for VHDL simulation, *Leonardo Spectrum* for the logical synthesis, Xilinx *Foundation* for place and route operations.

Table II summarizes main architecture features in terms of memory requirements, maximal running frequency, latency and processing times for different image sizes.

As mentioned before, the required memory space depends on the image length. Performance results show that this dimension has also an impact on the achievable maximal frequency. The explanation is that the input code is fully generic, and then the synthesis tool has chosen to distribute required memory into small LUTs of logical cells. In these conditions, a memory is made of a set of LUTs, and address decoding time is dependant on the number of LUTs. To avoid such limitations for the maximal running frequency, the solution can be to directly instantiate large blocks of memory in the VHDL code, but it would be at the expanse of its generecity.

	Image size	
	64x64	352x288
Internal memory(octets)	684	3 470
4 inputs LUTs for logical operations	1 166	2 463
Frequency(MHz)	45,8	33
Processing time (ms)	0,09	3,1
Latency time(μs)	18,6	75

TABLE II
SYNTHESIS RESULTS AFTER FINAL PLACE AND ROUTE

An image of size 100 Ko requires less than 4 Ko of internal memory to be compressed. Moreover, this image is totally

processed in 3 ms,which largely fulfils real-time constraints. The global latency time is 75 μs, and then can be considered extremely low for an image coder.

	Coder	
	JPEG	JPEG2000
Internal memoiry (octets)	3 100	82 000
4 inputs LUTs for logical operations	9 690	9 400
Frequency(MHz)	61	11

TABLE III
COMPARATIVE SYNTHESIS RESULTS FOR JPEG AND JPEG2000

Main architecture features for JPEG and JPEG2000 given in table III have been found respectively in [3] and [4], for minimal surface solutions. Comparative results clearly show that the LAR coder requirements for logical cells are much less than JPEG2000 and even than JPEG. From a memory point of view, LAR complexity is similar to JPEG one. The forthcoming integration of the Golomb-Rice entropy coder should have a limited impact on the logical resources. Only memory increase can become significant if we choose to realize k parameters learning on a larger range of values.

IV. CONCLUSION

We have presented in this paper a dedicated FPGA implementation of the FLAT LAR image coder. This coding technique is particularly well adapted for low bit rate compressions. From an image quality point of view, the FLAT LAR presents better results than JPEG, while implementation resources requirements are similar.

Internal architecture has been designed as a set of parallel and pipelined stages, enabling a full image processing during an unique regular scan. The architecture latency is extremely low as it is determined by the data acquisition for one slice of 8 lines.

Next developments will concern synthesis of Golomb-Rice codes as ultimate stage, as well as the design of the second LAR layer to encode local texture. This layer can be compared to JPEG technique adding variable block sizes representation, but in the same time substituting an integer transform (Hadamard) to DCT one.

REFERENCES

[1] "Jpeg-aic : scope and evaluation," *International Standards Organization working document, ISO/IEC SC29/WG 1/N4326*, 2007.

[2] A. Said and W. Pearlman, "Reversible image compression via multiresolution representation and predictive coding," in *Visual Communication and Image Processing*. SPIE, November 1993, vol. 209, pp. 664–674.

[3] Alma Technology, "Jpeg fast encoder," *www.alma-tech.com*, October 2002.

[4] O. Cantineau, "Real-time ip cores for jpeg2000 compression accelelation," *Bits and Chips Conference, Eidhoven*, April 2004.

[5] O. Déforges, M. Babel, L. Bédat, and J. Ronsin, "Color LAR codec : a color image representation and compression scheme based on local resolution adjustment and self-extraction region representation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 8, pp. 974–987, August 2007.

[6] M. Babel, O. Deforges, and J. Ronsin, "Interleaved S+P Pyramidal Decomposition with Refined Prediction Model," in *ICIP*, October 2005, vol. 2, pp. 750–753.

[7] H264 MPEG-4 10 AVC, "Joint committee draft (cd)," *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEGn 3rd Meeting : Fairfax, Virginia, USA*, May 2002.

[8] C.A. Shaffer and H. Samet, "Optimal quadtree construction algorithms," *Computer Vision, Graphics, Image processing*, vol. 37, October 1987.

[9] P. Strobac, "Tree-structured scene adaptive coder," *IEEE Trans. on Communication*, vol. 38, no. 4, April 1990.

[10] M. A. Losada and K. T. Mullen, "The spatial tuning of chromatic mechanisms identified by simultaneous masking," *Vision Res.*, vol. 34, no. 3, pp. 331–341, 1994.

[11] K. K. De Valois M. A. Webster and E. Switkes, "Orientation and spatial-frequency discrimination for luminance and chromatic gratings," *JOSA.*, vol. 7, no. 6, pp. 1034–1049, 1990.

[12] R. E. Graham, "Predictive quantizing of television signals," *IREWES-CON Conv. Rec.*, vol. 22, no. 4, pp. 147–157, 1958.

[13] P.G. Howard and J.S. Vitter, "New methods for lossless image compression using arithmetic coding," *Proc. of Data Compression Conference, IEEE*, pp. 257–266, 1991.

[14] G. Golomb, "Run-length encodings," *IEEE Trans. on Information Theory*, vol. IT-12, pp. 399–401, July 1966.

[15] R.F. Rice, "Some practical universal noiseless coding techniques," *Tech. Rep. JPL-79-22, JET Propulsion Laboratory, Pasadena, CA*, March 1979.

[16] T. Grandpierre, C. Lavarenne, and Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors," in *Proceedings of 7th International Workshop on Hardware-Software Codesign (CODES 99), Rome, Italy*, May 1999, pp. 74–78.