Efficient High-Performance ASIC Implementation of JPEG-LS Encoder

Markos Papadonikolakis[1], Vasilleios Pantazis[2] and Athanasios P. Kakarountas[3, 4]

[1] Alma Technologies, Pikermi Attica, Greece
[2] Desics Department, IMEC, Leuven, Belgium
[3] VLSI Design Lab., Electrical & Computer Engineering Department, University of Patras, Rio, Greece
[4] Department of Informatics on Biomedicine, University of Central Greece, Lamia, Greece

Abstract - *This paper introduces an innovative design which implements a high-performance JPEG-LS encoder. The encoding process follows the principles of the JPEG-LS lossless mode. The proposed implementation consists of an efficient pipelined JPEG-LS encoder, which operates at a significantly higher encoding rate than any other JPEG-LS hardware or software implementation while keeping area small.*

Index Terms - Image processing, lossless compression, JPEG-LS, LOCO-I, VLSI implementation.

I. INTRODUCTION

There are two main categories of image compression algorithms, the lossy and the lossless compression algorithms. The first category, which is more widely used, defines encoding processes that omit parts of the data to be encoded, exploiting the redundancy of the visual information. Hence the lossy algorithms present much higher ratios of compression than the lossless ones, which, as implied by their name, keep the data to be encoded intact. However the latter group of algorithms allows the complete reconstruction of the encoded image without loss of information, a property which lossy algorithms lack. Baseline JPEG is a very commonly used lossy compression algorithm and JPEG-LS, which is the targeted algorithm of this paper, is a typical lossless algorithm. In general, lossless algorithms are used in many specialized applications that emphasize more on the attainment of high fidelity and less on the compression ratio. These applications need to process images that may be satellite captured, medical or biometric. Furthermore, professional photography is a field in which lossless algorithms are able to thrive successfully.

The aforementioned group of applications presents a fundamental requirement. Large volumes of image data have to be compressed and transferred in real time. The latter disallows the use of software implementations, while in the same time renders the need for the development of hardware-implemented solutions imperative, as their advantages in terms of throughput, especially when considering an ASIC implementation, are widely known. Furthermore their operation is not impeded by other processes, something very common in the field of software implementations, which have to share their CPU time with other applications.

The JPEG-LS is an established standard for lossless and near-lossless compression of grayscale and color continuous-tone images [1]. It provides better compression ratios than the lossless JPEG standard, while maintaining a fairly low level of complexity. In the heart of the JPEG-LS lies the algorithm LOCO-I (Low Complexity Lossless Compression for Images) [2] and [3], a product of the collaboration of M. J. Weinberger and G. Seroussi (Hewlett Packard laboratories). The main feature of LOCO-I is that it combines the effective compression models proposed by context modeling with a low complexity level. This is achieved by the use of a combinational method that matches a simple coding unit to its modeling unit, while avoiding composite operations. Thus a low complexity instance of the universal context modeling is obtained. These attributes allow one of the highest compression ratios when compared to other lossless compression algorithms. Needless to say that most of those algorithms are also significantly more complex. However the potential of the JPEG-LS while in near-lossless mode is limited, and its performance in aspects of compression is not as good as that of other lossy compression algorithms (lossy JPEG). This does not affect the range of applicability of JPEG-LS in the fields of satellite or medical image compression, which require the reconstruction of the original information without alterations after the decoding process.

During the development process it was decided to propose a hardware implementation of JPEG-LS which implements the highly effective LOCO-I lossless compression mode and not the near lossless one. That decision presented two more advantages. First of all it reduces further the complexity of the algorithm, which leads to a smaller in terms of area and less energy-consuming integrated circuit. Apart from that by removing the elements of the LOCO-I algorithm that are used only in the near-lossless mode, the algorithm itself is simplified and its stages can be lessened. This can be exploited by applying various design techniques, leading to the reduction of the overall maximum delay that is necessary for the execution of the operations of the algorithm.

The designing process resulted in the ASIC implementation of an encoding unit which increases the throughput about five times when compared to other proposed ASIC implementations, while maintaining low area requirements in such a way that renders it suitable for use in portable communication devices. The main achievement of this work is the introduction of a real-time, small-sized, high-performance JPEG-LS lossless encoder, which maintains a wide range of

978-3-9810801-2-4/DATE07 © 2007 EDAA

applicability.

In the following sections of this paper we will delve into the operations of the algorithm and the architecture of the proposed hardware solution. The second section contains an overview of the JPEG-LS algorithm and a presentation of its most important functions. Section III analyzes the proposed JPEG-LS implementation in depth, providing details regarding its architecture, logic and any modifications made in order to decrease its critical path, as well as the prospective features of the integrated circuit. In Section IV the proposed JPEG-LS is implemented using ASIC technology. Its performance is compared to that of other implementations. Finally, in Section V the paper concludes.

II. THE JPEG-LS ALGORITHM

The JPEG-LS algorithm consists of two main units, a context modeler and an encoder. The context modeler uses the values of the adjacent to the current (the pixel to encode) pixels, correlates it to a specific context and predicts the current pixel value. The statistics of the selected context, which are based on past data, are used for the calculation of the prediction error and the coding variables. The statistic parameters are updated with the new information derived from the current pixel, and the coding unit encodes the prediction error using a Golomb-Rice code.

LOCO-I also uses a run mode, in order to compress constant regions. Run mode consists of a run scanner, which, when operation is set to run mode, checks if the current pixel is equal to the previous one; if so, run mode continues, else the modeler enters run interruption mode. Run coder either encodes the run length or the pixel that caused the run interruption. A closer look at the operations which comprise JPEG-LS follows.

A. Context Modeler

When in normal mode, the context modeler uses the values of the neighboring to the current pixels, in order to compute

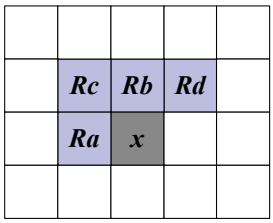

Fig. 1. Causal template.

the local gradients D1= Rd - Rb D2 = Rb - Rc and D3= Rc – Ra. The neighboring pixels Ra, Rb, Rc and Rd compose the "causal template" of the current pixel x, as illustrated in fig.1.

The local gradients are quantized and the modeler, using these three quantized vectors, selects one of 365 available contexts for the current pixel. Four statistic parameters are kept for each context. These consist of the accumulator of the magnitudes of previous prediction errors A[Q], the bias variable B[Q], which accumulates the errors' values, the prediction error correction parameter C[Q] and the occurrences counter N[Q].

The modeler uses a median edge detector to predict the value of the current pixel Px:

$$Px = \begin{cases} \min(Ra, Rb) & , \text{ if } Rc \geq \max(Ra, Rb) \\ \max(Ra, Rb) & , \text{ if } Rc \leq \min(Ra, Rb) \\ Ra + Rb - Rc & , \text{ otherwise.} \end{cases}$$

This operation is followed by the computation of the prediction error, which is corrected using the bias cancellation variable C[Q], and clamped in the appropriate range for the coding unit.

The final step before the encoding of the prediction error involves the calculation of the value of the Golomb variable k and the update of the current context statistic parameters. The derivation of k is performed using the parameters A[Q] and N[Q]:

```
for (k=0; (N[Q]<< k)< A[Q]; k++);
```

The prediction error is mapped to a non-negative value, with the use of variable k and the parameters B[Q] and N[Q], as Golomb codes can only be applied on positive values.

After the computation of Golomb variable k, the statistics are updated with the new prediction error. The bias correction parameter C[Q] is updated with at most one unit per iteration, according to the updated value of the bias variable B[Q]. This preserves the low complexity of LOCO-I.

B. Golomb Code

The operations of the context modeler are followed by the encoding of the mapped prediction error. This is performed using the optimal codes for a Two Sided Geometric Distribution (TSGD) [4], based on Golomb codes [3]. The encoding procedure is carried out as follows:

• If the number formed by all but the k least significant bits of the mapped error value is less than LIMIT – qbpp – 1, this number is encoded in the output bit stream in unary representation. This means that the number is represented with a number of zeros equal to its value, which are followed by a single '1'. Then the k least significant bits are appended to the output bit stream unaltered.

• If the aforementioned condition does not apply, a number of LIMIT – qbpp – 1 zeros and a single binary one is encoded in the output bit stream to represent a maximum length code. The entire mapped error value reduced by 1 is then encoded in the output bit stream unchanged, using qbpp bits. LIMIT and qbpp stand for the maximum number of bits of the Golomb code and the number of bits necessary to represent the mapped error value respectively. For 8-bit images, LIMIT is equal to 32 and qbpp to 8.

C. Overview

The LOCO-I block diagram (based on [2]) is illustrated in fig. 2. It is not difficult to notice the existence of a data-dependency loop in the algorithm flow. Every time that the context modeler determines that two subsequent samples belong in the same context, the previously updated parameter

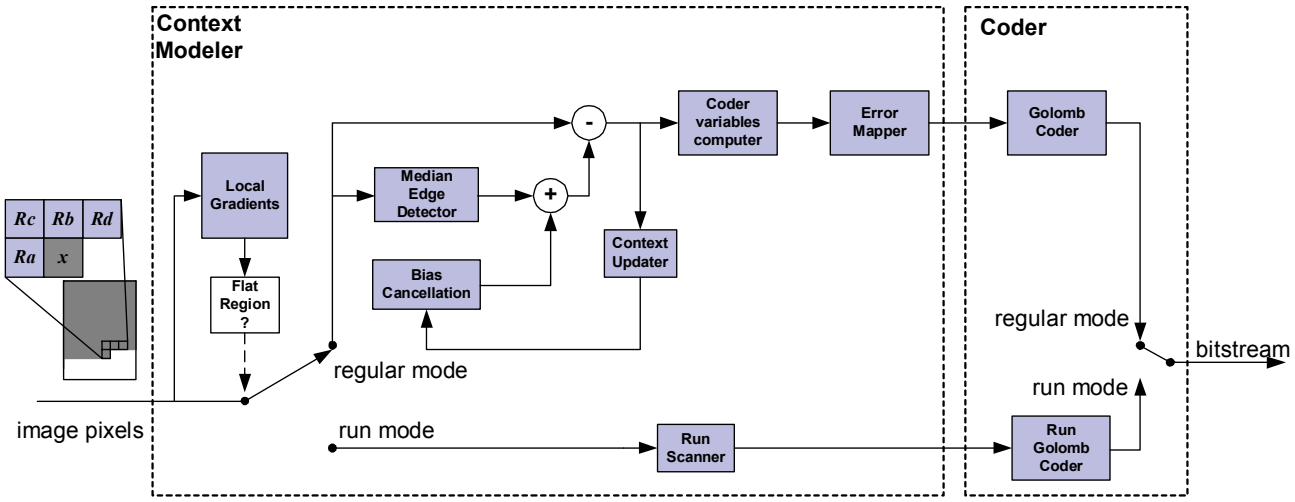

Fig. 2. LOCO-I block diagram (based on [2]).

C[Q] has to be used in order to perform the bias cancellation of the prediction error of the next sample. Thus the aforementioned loop includes the functions of the prediction error calculation and the statistic parameters' update. The statistic parameters can not be updated before the execution of the computations necessary to calculate the value of Golomb variable k and the mapping of the prediction error, operations that require the current values of A[Q], B[Q] and N[Q].

III. PROPOSED JPEG-LS IMPLEMENTATION

The main design technique used in this implementation is pipelining. It leads to a significant reduction of the critical path and thus to a high maximum throughput. It has been shown that the execution of the LOCO-I involves several functions and processes, from the context determination to the encoding of the mapped error. If an attempt to implement the algorithm's flow in a single stage was made, a critical path of unacceptable delay would be produced. This would limit extremely the maximum operating frequency. However, the distribution of the processes of the algorithm into more than one stage, via pipelining, minimizes the critical path and reduces the overall maximum delay significantly.

A. JPEG-LS Data Path

The most important aspect while breaking down the algorithm into stages is the placement of the pipeline registers. It is fundamental for delay-equivalent algorithmic phases to be produced. The placement of such registers after the context determination for the current sample, as well as before the encoding of the mapped error, is apparent as these processes are data-independent. The difficulty lies in separating the data-dependent processes such as the calculation of the prediction error and the update of the context statistic parameters. The calculation of the following prediction error needs the previously updated bias cancellation parameter C[Q] so pipelining can not be applied in this data-dependent loop. The critical path of the design is then observed in the data-loop circuit, including the prediction error calculation and the update of the statistic parameters. These functions are complex, resulting in a critical path whose delay would be higher than the desirable for a high-performance core.

Nevertheless, the low-complexity scheme of LOCO-I indicates that the update of the bias cancellation parameter C[Q] is limited to at most one unit per iteration. Thus, the next value of the bias parameter for the same context will be the same, reduced, or incremented by one. Since there are only three possible values for the updated parameter C[Q], the look-ahead technique can be applied to the computation of the error residual. The prediction error can be estimated for these three possible values and then stored in a pipeline register. At the next stage it will be possible to decide which one of the three estimated errors has to be used, by comparing the previous value of C[Q] with the updated one, as demonstrated in fig. 3.

The pipeline register's context update feeds the bias cancellation value to the prediction estimation only when two subsequent samples occur in the same context. Otherwise, C[Q] is provided by the context parameters' memory. It is necessary for that reason to have a double sized memory in order to store the values for bias cancellation C[Q], since two different values of bias cancellation have to be read from the memory in the same clock cycle: one for the error estimation and one for the parameters' update.

Apart from that, due to the fact that an extra clock cycle is required after the context determination in order to access the

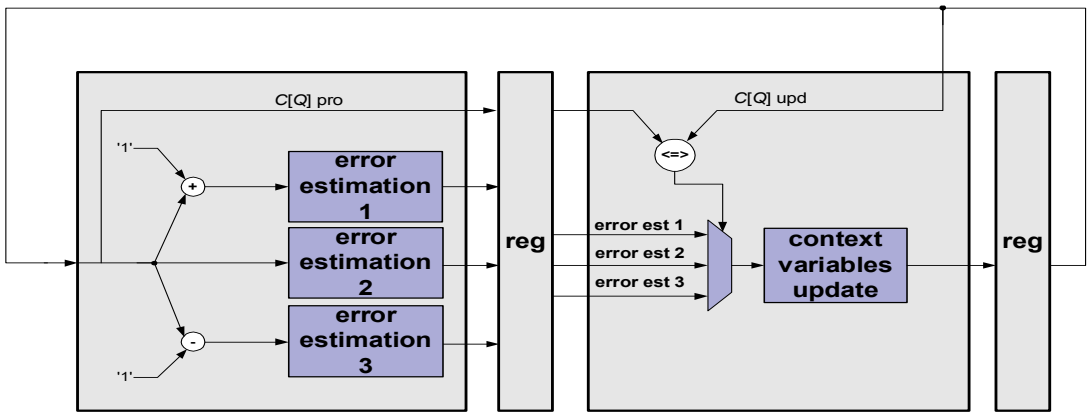

Fig. 3. Prediction error calculation with estimated bias cancellation.

selected context's parameters from the memory, it is possible to apply an additional pipeline stage before the error estimation phase. During the process of addressing the context memory several pre-computations which will speedup the error estimation of the next phase can be performed. For instance, Px can be subtracted from Ix beforehand, since both their values are already available (Ix and Px stand for the error and the Golomb variable k). Two pipeline stages have been used to implement the encoding process. The first stage produces the code for the current mapped error, while the second concatenates the code and produces the final encoded bitstream as its output. The calculations that need to be executed for the run mode scan can be implemented in the first pipeline stage, while determining the context of the current sample. The technique of pipelining has been used also in the implementation of the run interruption encoding procedures, even though those operations are simpler than when in normal mode, which makes their effect on the overall delay of the critical path negligible.

The data path's block diagram is illustrated in fig. 4. The first stage involves the determination of the context and computation of the prediction of the median edge detector. necessary to access the desired context parameters. The The pre-computations of the errors take place in the second stage, while the memory's decoding unit produces the address estimation of the prediction error is carried out in the third stage. It is done using the last updated value of C[Q]. In the fourth stage the selection of the correct estimated value of the prediction error is made. The next step involves the computation of the value of the Golomb variable and the mapping of the errors, while the context updater calculates the new values of the context statistic parameters. The mapped error is encoded in the two final pipeline stages.

The operations that take place in stage 4 are the ones that present the critical path of the design. In that stage the comparison between the previous and the updated value of C[Q] is carried out. It includes the multiplexer for the choice of the correct error estimation and also the parameters' update. The update is performed concurrently to the Golomb variable calculation and the error mapping, but its operations are slower.

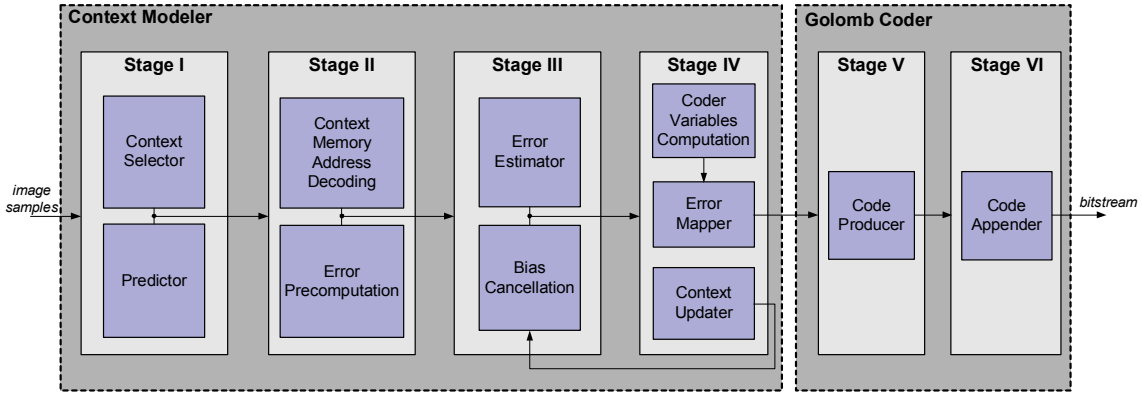

Fig. 4. LOCO-I's modified data path.

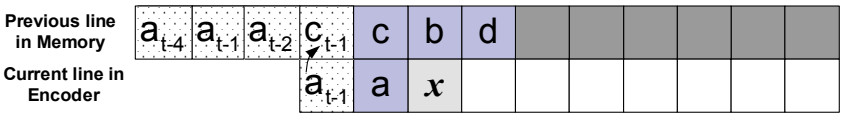

Fig. 5. Use of image line memory.

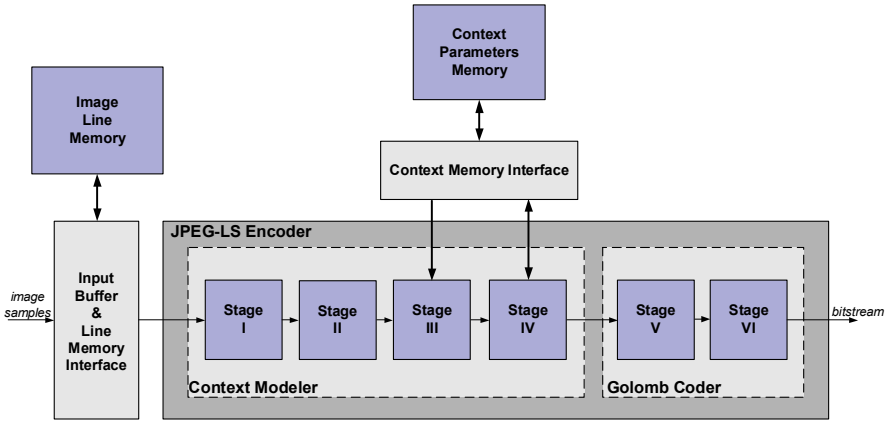

Fig. 6. Proposed JPEG-LS architecture.

B. Memory Requirements

The size of the required memory for the JPEG-LS encoding process depends on the storage of the context parameters and of the previously encoded samples. The C [Q] memory is double-sized, since two different addresses need to be accessed concurrently, in order for the operations of stage three and four to be carried out. The bit lengths of A[Q], B[Q], C[Q] and N[Q] for 8-bit images are 13, 7, 8 and 7, respectively. They are deduced from the possible range of their values. Thus the total size of the context memory is about 1.9 KB.

The minimum required memory needed to store the neighbor samples' values is one image line as demonstrated in fig. 5. Only the values of the neighbors a, b, c and d have to be accessed for the context determination of the current pixel x. A technique which allowed us to lessen the overall size of the memory is presented next. Sample x of instance t will become the neighbor a for the following image sample, thus it can be stored in a single register. The pixel c of t-1 instance will be no longer necessary as a neighbor so it can be replaced. Thus the memory size is minimized. Only one image line is required, while the works in [6] and [8] use a memory size of two image lines. This minimizes the total area of the ASIC design, since the on-chip memory consumes most of the area in such designs. The memories used are DUAL-RAMs, making it feasible to perform both read and write operations in the same clock cycle.

C. JPEG-LS Architecture

The architecture of the proposed JPEG-LS encoder is illustrated in fig. 6. It consists of the LOCO-I pipelined data, the context and the image line memory and two memory control interfaces. The line memory interface addresses the line memory, stores the new neighbor samples and feeds the LOCO-I data path with the values of the neighbors of the current sample. The context memory interface addresses the determined context parameters and feeds the error estimation stage with the last updated value of C[Q] as well as the fourth stage with the parameters of the context which was previously addressed. Finally the updated values are stored back in the context memory.

IV. Implementation and Results

The design was captured in VHDL and was fully simulated and verified using the Model Technology's ModelSim Simulator. The goal of this work was to propose a competitive hardware solution, so the design was synthesized for ASIC technology. Synthesis was performed in a Synopsys environment using several process libraries. The back-annotated information that was extracted from the synthesis tool was used to carry out the simulation of the design. The verification of the design's functionality and compatibility was based on the collection of test images [1] and on various popular testing images with the use of HP LOCO-I executable [5] as software benchmark.

The characteristics of the JPEG-LS implementation are demonstrated in table I, while the corresponding properties of other implementations available in academia are demonstrated in table II.

The authors have tried their best to be fair to the comparison with other works. Surprisingly, none of the available works gives a full report of the implementation's characteristics, in terms of area, operation frequency and throughput. Thus, the authors have assumed for some implementations that throughput is always the theoretic maximum and this is how table II was derived. At this point, it should be mentioned that implementation suggested by [7], also available in the scientific literacy, does not state its overall characteristics specifically.

Table III contains a comparison of the proposed TSMC 0.18μm implementation to other implementations available in academia. From the following table, it is easily observed that the proposed JPEG-LS implementation presents significantly higher throughput than any other available implementation, while minimizing the area requirements. This was expected due to the simplicity of the proposed implementation's algorithm and the application of pipeline stages at critical points of the encoder. It has to be remarked that the manipulation of the LOCO-I functions in a way that allowed us to produce a simpler design, of course with no effect on

the functionality of the algorithm, has offered significant gains in terms of throughput.

TABLE I
CHARACTERISTICS OF THE PROPOSED JPEG-LS IMPLEMENTATION FOR THE TARGETED ASIC TECHNOLOGIES

ASIC TECH.	LOGIC AREA (EQ. GATES)	MEM USAGE (BITS)	OPERATING FREQUENCY (MHZ)	THROUGHPUT (MPIXELS/S)
TSMC 0.18 μm	27681	23887 * 20815 **	183	183
UMC 0.18 μm	28127	23887 * 20815 **	160	160
TSMC 0.09 μm Sage-X Artisan	25709	23887 * 20815 **	265	265

* For 1024 pixels per image line
** For 640 pixels per image line

TABLE II
CHARACTERISTICS OF OTHER JPEG-LS IMPLEMENTATIONS

WORK	TECH.	LOGIC AREA (EQ. GATES)	MEM USAGE (BITS)	OPERATING FREQUENCY (MHZ)	THROUGHPUT (MPIXELS/S)
[6] *	-	49,457 gates	32768 (1024)	66	66 upper limit (theoretic)
[8] *	0.18 μm	70,000 gates	24000 (640)	40	40 upper limit (theoretic)

* Implementing lossless and near-lossless mode

TABLE III
COMPARISONS WITH THE OTHER JPEG-LS IMPLEMENTATIONS

WORK	COMPARED TO THE PROPOSED AREA (%)	COMPARED TO THE PROPOSED MEMORY (%)	COMPARED TO THE PROPOSED THROUGHPUT (%)
[6]	178.7 %	137 %	36.1%
[8]	252.9 %	115 %	21.8%

Table III reveals that the proposed implementation is by far the most efficient in terms of throughput as it presents almost five times higher throughput than the other ASIC implementations. Also its requirements in terms of area are significantly lower, being at least 1.7 times less than those of other ASIC implementations.

V. CONCLUSION

This paper presented a novel implementation of a lossless JPEG-LS hardware encoder. The pipelining design techniques used allowed the full exploitation of the low-complexity features of the LOCO-I algorithmic functions leading to a very high operating frequency, the highest ever reported for a JPEG-LS encoder in academia or industry. The proposed implementation presents up to five times higher throughput than any other available ASIC JPEG-LS implementation.

Apart from its outstanding performance in speed, the proposed design approach is quite efficient in terms of area, where its requirements are very low. Finally, as it was manifested in the section containing the results of the implementation, the proposed encoder is very cost-effective while being fully defined in terms of area, operation frequency and throughput.

ACKNOWLEDGMENTS

We would like to thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the program PYTHAGORAS, for funding the above work.

REFERENCES

[1] Information Technology-Lossless and near-lossless compression of continuous-tone images-Baseline. International Telecommunication Union (ITU-T Recommendation T.87). ISO/IEC 14495-1, 1998.

[2] M.J. Weinberger, G. Sapiro and G. Seroussi, "The LOCO-I Lossless image compression algorithm: Principle and standardization into JPEG-LS", IEEE Trans. on Image Processing, vol. 9, Aug. 2000, pp. 1309-1324.

[3] S. W. Golomb, "Run-length encodings", IEEE Trans. Inform. Theory, vol. IT-12, pp. 399–401, July 1966.

[4] M. J. Weinberger, and G. Seroussi, "Optimal prefix codes for sources with two-sided geometric distributions," IEEE Trans. Inform. Theory, vol. 46,pp.121–135, Jan.2000.

[5] M. J. Weinberger, and G. Seroussi, "From LOCO-I to the JPEG-LS Standard", Computer Systems Laboratory, HP Laboratories Palo Alto, HPL-1999-3, January 1999. http://www.hpl.hp.com/loco/

[6] A. Savakis and M. Pioriun, "Benchmarking and Hardware Implementation of JPEG-LS", Rochester, NY, Sept. 2002, International Conference on Image Processing Proceedings (ICIP '02), Volume 2 pp II-949- II-952.

[7] M Ferretti, M. Boffadossi, "A Parallel Pipelined Implementation for JPEG-LS", Dip. Informatica e Systemistica, Univ. Pavia, Italy, 17th International Conference on Pattern Recognition (ICPR'04) - Volume 1 pp. 769-772.

[8] Xiang Xie, GuoLin Li and ZhiHua Wang, "A Near-lossless Image Compression Algorithm Suitable for Hardware Design in Wireless Endoscopy System", Department of Electronic Engineering, Tsinghua University, Beijing, P. R. China, 100084, ASICON 2005. 6th International Conference On ASIC, Volume 1, 24-27 Oct. 2005 pp. 37 – 40.