ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.14, No.2, 1–6

# Scalable spatio-temporal parallel parameterizable stream-based JPEG-LS encoder

**Cássio A. Carneiro**[a)], **Francisco P. Garcia, Henrique C. Freitas, Carlos P. S. Martins, and Flávia M. F. Ferreira**

*Pontifical Catholic University of Minas Gerais,*

*Av. Dom Jose Gaspar, 500, Belo Horizonte, MG, Brazil*

a) *cassio@unifemm.edu.br*

**Abstract:** A parallel parameterizable stream-based JPEG-LS encoder architecture for scalable throughput is presented. The main contribution is the reconfigurable spatio-temporal parallelism to meet different pixel rates for lossless video compression, achieving the required throughputs with lower processing frequencies, scaled by the degree of spatial parallelism. The proposal allows for optimized performance associated to device resources and processing frequency. Experimental results were verified in Altera Stratix I FPGA device.

**Keywords:** JPEG-LS, lossless video compression, scalable throughput, FPGA, spatio-temporal parallelism, reconfigurable architecture

**Classification:** Integrated circuits

## References

[1] ISO/IEC: Information technology - Lossless and near-lossless coding of continuous tone still images - Baseline, International Telecomunications Union (ITU-T Recommendation T.87), 14495-1 (1998).

[2] M. Ferretti and M. Boffadossi: "A parallel pipelined implementation of LOCO-I for JPEG-LS," ICPR'04 17th International Conference on Pattern Recognition (2004) 769 (DOI: 10.1109/ICPR.2004.1334311).

[3] P. Merlino and A. Abramo: "A fully pipelined architecture for the LOCO-I compression algorithm," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **17** (2009) 967 (DOI: 10.1109/TVLSI.2008.2009188).

[4] M. E. Papadonikolakis, *et al.*: "Efficient high-performance implementation of JPEG-LS encoder," J. Real-Time Image Process. **3** (2008) 303 (DOI: 10.1007/s11554-008-0088-7).

[5] J. Lei, *et al.*: "A new pipelined VLSI architecture for JPEG-LS compression algorithm," Proc. SPIE **7084** (2008) 70840O (DOI: 10.1117/12.794543).

[6] W. Wei, *et al.*: "Onboard optimized hardware implementation of JPEG-LS encoder based on FPGA," Proc. SPIE **8514** (2012) 851406 (DOI: 10.1117/12.930869).

[7] B.-S. Kim, *et al.*: "A high performance fully pipeline JPEG-LS encoder for lossless compression," IEICE Electron. Express **10** (2013) 20130348 (DOI: 10.1587/elex.10.20130348).



DOI: 10.1587/elex.14.20160950
Received September 29, 2016
Accepted November 21, 2016
Publicized January 6, 2017
Copyedited January 25, 2017

[8] Z. Wang, *et al.*: "A high performance fully pipelined architecture for lossless compression of satellite image," ICMT International Conference on Multimedia Technology (2010) 1 (DOI: 10.1109/ICMULT.2010.5631429).
[9] L.-J. Kau and S.-W. Lin: "High performance architecture for the encoder of JPEG-LS on SOPC platform," IEEE Workshop on Signal Processing Systems (SiPS) (2013) 141 (DOI: 10.1109/SiPS.2013.6674495).
[10] G. Schaefer, *et al.*: "An evaluation of lossless compression algorithms for medical infrared images," IEEE-EMBS 27th Annual International Conference of the Engineering in Medicine and Biology Society (2005) 1673 (DOI: 10.1109/IEMBS.2005.1616764).

## 1 Introduction

JPEG-LS is an image compression standard that aims at outperforming lossless JPEG. The core algorithm behind it is called LOw COmplexity LOssless COmpression for Images (LOCO-I) [1], which provides low power consumption and storage facilities, making JPEG-LS ideal for hardware implementation for embedded systems [2, 3, 4, 5]. However, LOCO-I presents a sequential encoding algorithm that hinders the parallel implementation of JPEG-LS, which is expected to achieve high throughputs of pixels. Recent works [6, 7, 8, 9] have proposed pipelined architectures in order to overcome this drawback, but they only achieved throughputs equal to the internal frequency of the processing unit. The highest reported throughput was 120 Mpixels/s at a processing frequency equal to 120 MHz at the FPGA [7]. Though, there are lossless video compression applications that require higher pixel throughputs, such as high-definition TV, digital cinema and medical imaging, among others [8, 10]. Using the presented architectures, this requirement can only be reached by increasing the processing frequency. Therefore, literature still lacks hardware architectures that enable achieving throughputs higher than the internal FPGA processing frequency, in order to be able to meet increasingly stronger requirements.

## 2 The proposed architecture

The major requirement of the proposed architecture is that the complete system must be implemented on a single FPGA device, without external memory, aiming at fast manufacturing, as well as low-cost and low-energy-consumption circuits. To attend this, no image frame can be stored. This is the basis for scalable circuits intended to meet requirements of data throughput suitable to different classes of applications, with no need to change the memory device.

With the objective of providing the addressed requirements, this architecture employs reconfigurable spatial parallelism based on Single Instruction Multiple Data (SIMD) model, mapping N Processing Units (PUs) to encode different image partitions. Here, N can be understood as the degree of spatial parallelism. Each PU makes use of time parallelism, aware of the inherent sequential nature of LOCO-I. Encoding the current X pixel requires five context pixels, which belong to the current row (Ra and X) or to the previous one (Rc, Rb and Rd), as shown in Fig. 1.


DOI: 10.1587/elex.14.20160950
Received September 29, 2016
Accepted November 21, 2016
Publicized January 6, 2017
Copyedited January 25, 2017

Thus, only two lines of pixels are previously stored to enable spatial parallelism of all the PUs, which work synchronously, whereas the architecture is fed by a single video data stream. One of the rows (the last one received) is necessary for changing the clock domain, allowing each PU to operate at a processing rate corresponding to 1/N of the input pixel rate (Clk_px). The second row is used to store the pixels in the previous row that are also necessary to encode the X pixel. Fig. 1 shows 4 PUs being used to process distinct partitions of a video frame.

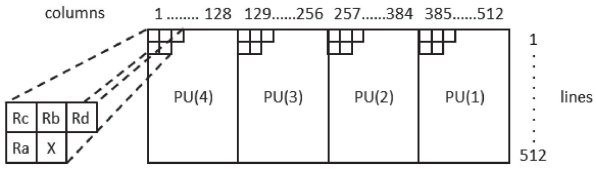


**Fig. 1.** Image's partitioning for spatial parallelism.

Since JPEG-LS is a context-dependent method and each partition presents a context different from the one of the whole image, partitioning should influence the overall compression rate. So, tests were performed in MATLAB to evaluate this impact, using, as in [10], the implementation of the University of British Columbia. Table I shows results (in bits per pixel) achieved with the parallel architecture on 8-bit test images with different kinds of partition. For the sake of clearness, the best compression rate for each image is bolded. As results show, the partitioning has little effect on compression rate. The partition into two and into four regions improved, on average, 0.37% and 0.43%, respectively, compared to the image with no partitioning.

**Table I.** Bits per pixel for 1, 2 and 4 vertical partitions

| Image | $1 \times 512 \times 512$ | $2 \times 256 \times 512$ | $4 \times 128 \times 512$ |
|---|---|---|---|
| Barbara | 4.7326 | 4.7059 | **4.7057** |
| Peppers | 4.3033 | **4.2932** | 4.3064 |
| Lena | 4.2557 | **4.2448** | 4.2461 |
| Sailboat | **4.7807** | 4.7829 | 4.7827 |
| Goldhill | 4.7199 | **4.7193** | 4.7421 |
| Cameraman | 4.8522 | 4.7795 | **4.7119** |
| Baboon | 5.8198 | **5.8169** | 5.8251 |

Fig. 2 shows the functional block diagram of the proposed parameterizable parallel architecture for the JPEG-LS encoder, applied to 8-bit images. The first level is composed of two cooperative subsystems: Data Processing Module and Control Module. The second level, depicted in Fig. 2a, is constituted of the structural parts of the Data Processing Module: the Input Interface Unit, the Mask Storage Units (MSUs), the Processing Units (PUs) and the Output Interface Unit. The PUs operate on the input pixels and each one is a complete LOCO-I encoder. The parameter N is the number of MSUs and PUs and it is configurable in synthesis time at the parallel arrangement in order to meet the throughput requirement. So,


DOI: 10.1587/elex.14.20160950
Received September 29, 2016
Accepted November 21, 2016
Publicized January 6, 2017
Copyedited January 25, 2017

while the input pixels are fed into the Input Interface Unit, a (1 × N) DEMUX distributes the pixels to be processed to each of the N MSUs. The Control Module, in turn, is in charge for controlling the Data Processing Module operations.

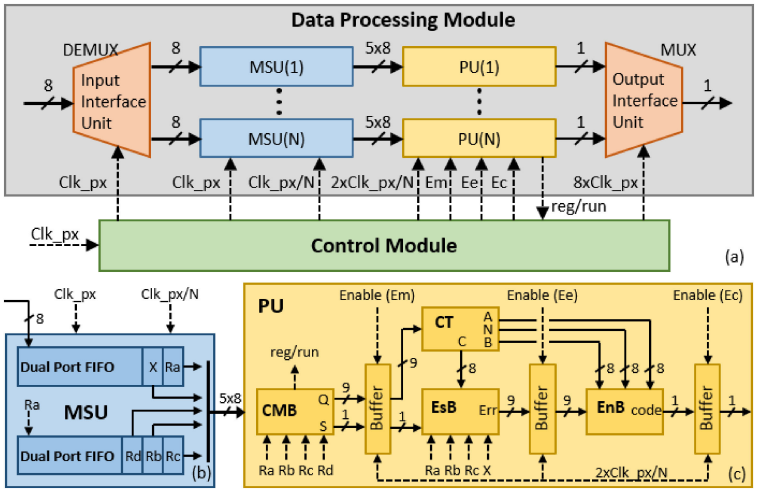


**Fig. 2.** JPEG-LS encoder: (a) Control and Data Processing Modules, (b) MSU blocks, (c) PU blocks.

The third level is represented by blocks that perform the MSU and the PU (portrayed, respectively, in Fig. 2(b) and Fig. 2(c)). The MSU acts as a storage device that provides its corresponding PU simultaneously with the five context-pixels required for encoding a specific X pixel. To suit this requirement, the implementation of each MSU uses: (a) one FIFO (First-In-First-Out memory) and two registers to store the pixels of the current line (this set is referred to as *Domain clock FIFO* in Fig. 3); and (b) one FIFO and three registers to store the pixels of the previous line (this set is referred to as *Line delay FIFO* in Fig. 3). Since the number of pixels to be stored at each MSU decreases as N increases, the total allocated memory is constant, independent on the value of N to meet the processing demand.

Fig. 3 shows writing and reading timing diagrams at the MSUs. When a rising edge of H_sync informs the beginning of a video line, the *Domain clock FIFO* of the MSUs sequentially store 1/N of the data row (Wr_MSUi, when high, activates the writing operation at MSUi). At each MSU, the output of the last register of the *Domain clock FIFO* is the input to its corresponding *Line delay FIFO*. At the beginning of the reception of the next row of pixels (signaled by a new period of H_sync), the five context-pixels are read simultaneously by the PUs from the MSUs, with an initial latency corresponding to a complete row of pixels (Rd_MSUi, when high, activates the reading operation at MSUi). Note that at *Domain clock FIFO*, the writing clock is the pixel clock (Clk_px) and the reading clock is Clk_px/N, which is also used as both writing and reading clocks at *Line delay FIFO*.


DOI: 10.1587/elex.14.20160950
Received September 29, 2016
Accepted November 21, 2016
Publicized January 6, 2017
Copyedited January 25, 2017

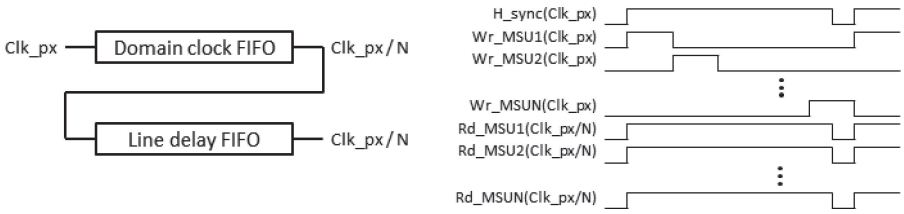


**Fig. 3.** Writing and reading timing at the MSUs

Each PU receives the five context pixels of its partition as parallel inputs from its corresponding MSU and then yields the pixel of the compressed video stream as serial output, i.e., one bit at a time. The Output Interface Unit has N 1-bit inputs and one 1-bit output. It is responsible for concatenating the serial data received from the PUs and transmit them in a single bit stream.

The Context Modeling Block (CMB), portrayed in Fig. 2(c), models the context Q of the current pixel [1], represents it with nine bits and updates the Context Table (CT) for context variables A(Q), N(Q), B(Q) and C(Q). Signal S indicates if a reversion of the context Q has taken place. The CMB also determines the type of encoding (regular or run) and informs it to Control Module.

The Estimating Block (EsB) performs context-based estimation and requires that context updating relative to the previous pixel is complete. The Golomb Rice Code is used to compress the residual error (Err) at Encoding Block (EnB), which stores data serially in a $1024 \times 1$ FIFO.

The JPEG-LS encoding presents a drawback named data-dependence, whenever consecutive pixels belong to the same context. In this case, encoding the current pixel depends on the context update of the previous pixel being complete. This problem can be addressed in several ways, as can be found in [2, 3, 4, 7, 8, 9]. In this present work, the solution adopted was to implement time parallelism by the pipelined structure shown in Fig. 2(c), where the blocks of PU operate at frequency $CLK_{PU}$ (MHz), which is twice the PU's input clock (i.e., twice Clk_px/N).

## 3 Results

The proposed architecture was verified on Altera Quartus-II software, targeted to Stratix EP1S10F484C5 FPGA device. Table II summarizes results of the synthesized circuit with one, two, four and eight PUs (number limited only by the amount of logical resources of target FPGA). $CLK_{PU}$ (MHz) represents the maximum clock frequency used by the PU's buffers; LE quantifies the necessary amount of Logic Elements and $T_{max}$ (Mpixels/s) is the achievable pixel throughput. In the first three lines of Table II the synthesis tool used speed optimization. In the last line, due to the proximity of the device's total occupation, the synthesis tool used area optimization, causing a relative worsening in speed.

**Table II.** Results for parallel JPEG-LS encoder prototypes

| N | $CLK_{PU}$ (MHz) | LE | FPGA occupation | $T_{max}$ (Mpixels/s) |
|---|---|---|---|---|
| 1 | 39.39 | 1277 | 12% | 19.69 |
| 2 | 38.25 | 2553 | 24% | 38.25 |
| 4 | 38.09 | 5104 | 48% | 76.18 |
| 8 | 30.29 | 9241 | 87% | 121.16 |


DOI: 10.1587/elex.14.20160950
Received September 29, 2016
Accepted November 21, 2016
Publicized January 6, 2017
Copyedited January 25, 2017

It can be noticed that the requirement that PU's processing frequency $CLK_{PU}$ is twice the PU's input clock poses a problem that is quite significant only if a single PU is used. In that case, the pixel throughput $T_{max}$ supported is only half the processing frequency $CLK_{PU}$. However, as N increases, $T_{max}$ raises proportionally. Table III shows the number of frames per second (fps) that can be processed by the implemented architecture at different frame resolutions, according to N.

**Table III.** Frame processing rate (fps) at different resolutions

| Resolution | 1 PU | 2 PUs | 4 PUs | 8 PUs |
|---|---|---|---|---|
| 320 × 240 | 256 | 498 | 991 | 1577 |
| 640 × 480 | 64 | 124 | 247 | 394 |
| 800 × 600 | 41 | 79 | 158 | 252 |
| 1024 × 768 | 25 | 48 | 96 | 154 |
| 1280 × 1024 | 15 | 29 | 59 | 92 |

All the related works did not use spatial parallelism, requiring a unitary Throughput Scale (TS), defined as the ratio between throughput and processing frequency ($CLK_{PU}$). As example, in [7], a throughput of 120 Mpixels/s was achieved at 120 MHz and higher processing frequencies would be required to increase pixel throughput. Though, gains on throughput based on frequency hardly overcomes barriers of constraints on either energy consumption or technology. Differently from the related works, in our proposal a scalable throughput is derived from changing the TS to N/2. Gaining benefits from the reconfigurable spatial parallelism, the architecture is squarely suited to achieve very different requirements on pixel throughput.

## 4 Conclusion

This paper highlights an efficient parallel parameterizable reconfigurable architecture for JPEG-LS encoder based on temporal and spatial parallelisms. The temporal parallelism, as in other related works, is implemented as pipelines in the context of a single PU and aims at breaking dependencies regarding the sequential nature of the JPEG-LS in order to improve throughput.

However, the main contribution is the reconfigurable spatial parallelism that yields higher throughput by increasing N, without rising the processing frequency $CLK_{PU}$, at the expense of increased use of FPGA logical resources. Thus, with spatial parallelism the achieved throughput is $T = (N/2)CLK_{PU}$.


DOI: 10.1587/elex.14.20160950
Received September 29, 2016
Accepted November 21, 2016
Publicized January 6, 2017
Copyedited January 25, 2017