Noname manuscript No.
(will be inserted by the editor)

MPCM: A hardware coder for super slow motion video sequences

Estefanía Alcocer · Otoniel López-Granado · Roberto Gutierrez · Manuel P. Malumbres

Received: date / Accepted: date

Abstract In the last decade the improvements in VLSI integration levels and image sensor technologies has leaded to a frenetic rush to provide image sensors with higher resolutions and faster frame rates. As a result, video devices were designed to capture real-time video at high resolution formats with frame rates reaching the 1000 fps and beyond. This ultra high-speed video cameras are widely used in scientific and industrial applications like car crash tests, combustion research, materials research and testing, fluid dynamics, flow visualization, etc. that demand real-time video capturing at extremely high frame rates with high definition formats. Therefore, data storage capability, communication bandwidth, processing time and power consumption are critical parameters that should be carefully considered in their design. In this paper, we propose a fast FPGA implementation of a simple codec called MPCM that reports similar quality than traditional PCM coding scheme but is able to reduce the bandwidth requirements up to 1.4 times, permitting current high-speed cameras to capture in a continuous manner into a massive storage device like an SSD.

Keywords PCM · image coding · FPGA design · high speed · integrated circuits

This research was supported by the Spanish Ministry of Education and Science under grant TIN2011-27543-C03-03

E. Alcocer O. López-Granado M. P. Malumbres
Physics and Computer Architecture Department
Miguel Hernández University. Elche, Spain 03202
Tel.: +34-966658392
E-mail: estefania.alcocer@goumh.umh.es,{otoniel,mels}@umh.es

R. Gutierrez
Communications engineering Department
Miguel Hernández University. Elche, Spain 03202
E-mail: roberto.gutierrez@umh.es

1 Introduction

Video compression has been an extremely successful technology that has found its commercial application across many areas, from scientific and industrial applications like video archiving, high quality medical video, surveillance and security applications, to the audio-visual industry (TV and cinema) and the broad spectrum of video appliances available in the market such digital cameras, DVD, Blue-Ray, DVB, etc.

In the last decade the improvements in VLSI integration levels and image sensor technologies has leaded to a frenetic rush to provide image sensors with higher resolutions and faster frame rates. As a result, video devices were designed to capture real-time video at high resolution formats with frame rates above 100 Hz. Nowadays, it can be found in the market ultra high-speed video cameras like Phantom v641 [11] which is able to capture high resolution video (2560 x 1600 pixels) at 1450 frames per second (fps). These video cameras are specially suited for scientific or industrial applications like car crash tests, explosives and pyrotechnics, ballistics, projectile tracking, combustion research, materials research and testing, fluid dynamics, flow visualization, etc. which demand real-time video capturing at extremely high frame rates with high definition formats. Therefore, data storage capability, communication bandwidth, processing time and power consumption are critical parameters that should be carefully considered in the design process of high-speed video camera.

To fight against these constraints, most of nowadays high-speed cameras store the captured images in a fast SDRAM module of up to 64GB [16, 5, 7, 11] without performing compression, using Pulse Code Modulation (PCM) [9]. The huge amount of data of the resulting uncompressed image/video needs to be processed to guarantee its transmission or storage, being a really challenging task. Thus, the internal communication bus may not be fast enough to transfer the video out of the camera, or the writing speed of the storage device may not be high enough to save the video [6]. So, the approach of using fast SDRAM memory as video storage is feasible since memory bandwidth is high enough, but when memory is run out, the camera stops recording and needs to save the stored video to a secondary storage in raw or compressed format. This is a limitation because depending on the capturing resolution of the camera, only a few seconds could be recorded in the RAM module, and so, continuous capturing is not possible.

In order to overcome these restrictions, it would be of interest to reduce the video storage requirements by means of hardware encoders that fulfill the application requirements, i.e. high frame rate and high definition and beyond video formats. Therefore, if we are able to perform some kind of ultra fast encoding, we would reduce the required storage resources, and real-time recording like in conventional video cameras would be possible.

Many hardware coders based on different coding algorithms are used in real systems [18, 8, 2, 13, 17, 1, 14, 4]. Most of them are Application Specific Integrated Circuits (ASICs) dedicated to specific encoding algorithms that are

not designed to work in real-time with ultra high frame rates and high definition video formats.

However, several attempts have been made in order to deal with high-speed camera encoding. In [3] authors present JPEG FPGA-based encoder which is able to compress up to 500 frames/s at a resolution of 1280x1024. Also, in [19] an improved version of the Fast Boundary Adaptation Rule (FBAR) [10] algorithm in conjunction with Differential Pulse Code Modulation (DPCM) is applied to increase the R/D efficiency, although coding delays were not provided.

In general, the constraints imposed by ultra-high frame rate video capture applications discard most of the existing coding techniques (e.g., predictive coding or transform coding) since they are much more complex than PCM and/or use predictive and entropy coding (which destroys the scalability and random access properties). Therefore, a coding algorithm that has properties similar to those of PCM (low complexity, random access, and scalability) but with a better coding efficiency would be of interest. Modulo-PCM (MPCM) [12] image coder fulfills these requirements. To encode an image, MPCM encoder removes certain bits from each pixel value which represents a very simple processing. The complexity is moved to the decoder side, where the bits that were removed from each pixel will be predicted by using its codeword (remainig bits of a pixel) and side information (SI) that the decoder computes by interpolating previously decoded pixels.

In this paper we implement a fast codec based on Modulo Pulse Code Modulation scheme (MPCM) [12] over a XC7Z020-1CLG484CES Xilinx FPGA device. Results show that FPGA-based MPCM encoder obtains a throughput of up to 409.84 MBytes/sec at high compression rates, allowing to store on a non volatile memory 2501 frames per second at a 1280 x 1024 resolution. Furthermore, in this paper we present a hardware implementation of the MPCM decoding system, which is able to reproduce a Full-HD definition video at 204 frames per second.

The rest of the paper is organized as follows. In section 2 we present a brief overview of the Modulo-PCM encoder. In section 3, the description of the proposed architecture is presented. A detailed evaluation of the architecture proposal is shown in section 4 in terms of R/D, coding delay, power consumption and occupied board area. Finally, in section 5 some conclusions are drawn.

2 Encoding system

In this section we describe the MPCP-based coding algorithm for the encoding of a one-dimensional signal. Let $\hat{x}_n$ $(n \in N)$ be a continuous-amplitude discrete-time signal whose amplitude values lie in $[A_{min}, A_{max}]$. Let x_n be the digital signal that results from the quantization and encoding of $\hat{x}_n$ with a fixed-rate uniform quantizer of B bits/sample and step-size:
$\Delta = (A_{max} - A_{min})\,/2^B$.

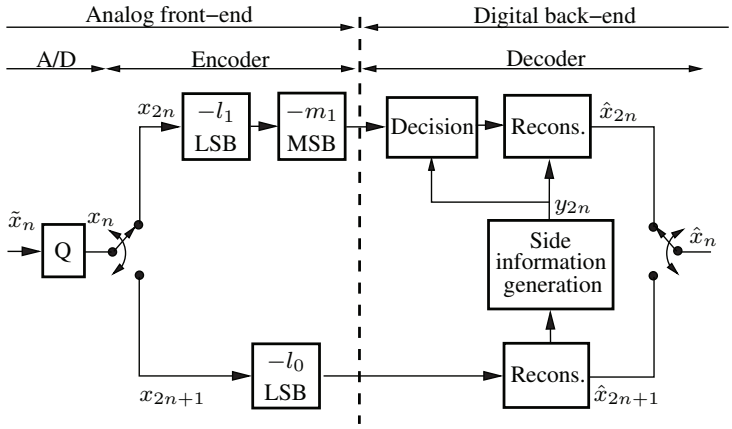

Fig. 1 Block diagram of MPCM coding algorithm

The easiest way of reducing the bit rate of $\hat{x}_n$ is to remove the l-LSBs (Least Significant Bits) of each codeword of $\hat{x}_n$. To achieve a more efficient rate reduction, we propose the use of a MPCM-based coding algorithm (Fig. 1). The samples of $\hat{x}_n$ are divided into sets that are encoded with different accuracies. For the sake of simplicity, let us consider we divide $\hat{x}_n$ into two sets: $S_0 = \{x_{2n+1}|n \in i = 0, 1, 2, \ldots\}$ and $S_1 = \{x_{2n}|n = 1, 2, \ldots\}$. As shown in Fig. 1, each sample in S_0 is encoded by removing the l_0-LSBs of its codeword while each sample in S_1 is encoded by removing the m_1-MSBs (Most Significant Bits) and l_1-LSBs of its codeword. Then, the encoder works at an average rate $R = B - (l_0 + l_1 + m_0)/2$ bits/sample.

As the encoding of x_{2n+1} is equivalent to a quantization with a uniform quantizer with step size equal to $2^{l_0}\Delta$, the decoder can directly reconstruct the samples of S_0 from their codewords (Fig. 1). With respect to the encoded samples in S_1 (Fig. 2(a)), removing the l_1-LSBs of x_{2n} is equivalent to quantizing its original continuous value with a uniform quantizer with step size equal to $2^{l_1}\Delta$ (Fig. 2(b)). After removing the m_1-MSBs, the resulting codeword identifies a set of 2^{m_1} disjoint intervals $\{I_i|i = 0, \ldots, 2^{m_1} - 1\}$, with each interval being of length $2^{l_1}\Delta$ (Fig. 2(c)).

In the decoder, to decide which I_i interval $\hat{x}_{2n}$ belongs to, MPCM decoders exploit the correlation between the signal samples by furnishing a prediction for each sample in S_1 based on previously decoded samples in S_0. This prediction y_{2n} acts as SI (Side Information) to decide the interval (Fig. 2(d)). The accuracy of the SI depends on the degree of correlation between the samples x_n and the distortion introduced in the encoding of S_0. To limit the impact of the coding distortion on the quality of the SI, in the encoding algorithm l_0 must be lower than l_1 ($l_0 < l_1$). If the decision process is done without error for x_{2n}, its m_1-MSBs are properly recovered. Once the decoder has estimated the m_1-MSBs of x_{2n}, it tries to recover its l_1-LSBs which finally provides a

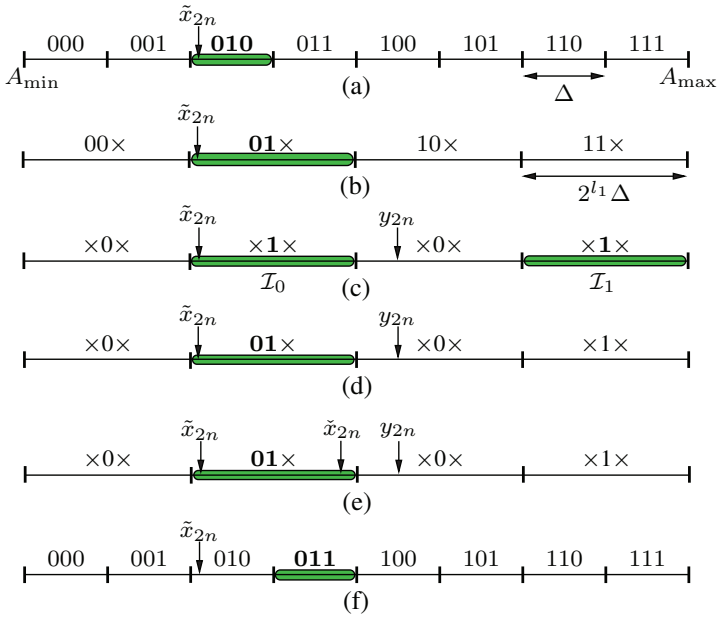

Fig. 2 Quantization intervals and codewords for R = 3, $l_1 = 1$ and $m_1 = 1$: (a) after A/D conversion, (b) after removing l_1 bits, (c) after removing m_1 bits, (d) after deciding between I_0 and I_1, (e) after reconstruction and (f) after final quantization. Symbol X represents the removed bits. Boldfaced codewords represent the codeword selected in each step for the values of x_{2n} and y_{2n} shown. Marked intervals are the intervals represented by the selected codeword in each step

reconstruction $\hat{x}_{2n}$. This reconstruction is done using the quantization interval I_i where supposedly lies x_{2n} and its SI (y_{2n}).

For a more detailed description of MPCM encoder, the reader is referred to [12,15].

3 Hardware Implementation

In order to cope with the high throughput bandwidth of nowadays high speed cameras, both MPCM encoder and decoder have been implemented over a hardware architecture. The description language used to build its design is VHDL. The proposed hardware implementation has been developed over a Zynq-7000 FPGA of Xilinx family, specifically over the ZC702 model which includes the XC7Z020-1CLG484CES SoC (System-on-Chip) [20].

3.1 Encoder architecture

The implemented encoder architecture is illustrated in Fig. 3. The original image captured by the camera sensors is stored in the memory block whose

reading is determined by the control block. In that structure, 16 pixels are read on each cycle so as to speed the encoding process as much as possible within the scope of the device's internal memory. This memory acts as an internal buffer of frames to read them in high speed applications and it is implemented with block RAMs. In this way, we use 52 Dual-port 36 Kb block RAMs with ports configured as $512x64$ bits, where 64 output bits, namely 8 pixels, are read in each memory output port. These pixels are processed in the following block, without delay, in the same reading cycle, where they are encoded by removing the corresponding l_0 or l_k and m_k bits. Finally, we obtain the coded samples of the image which are sent to the final storage device.

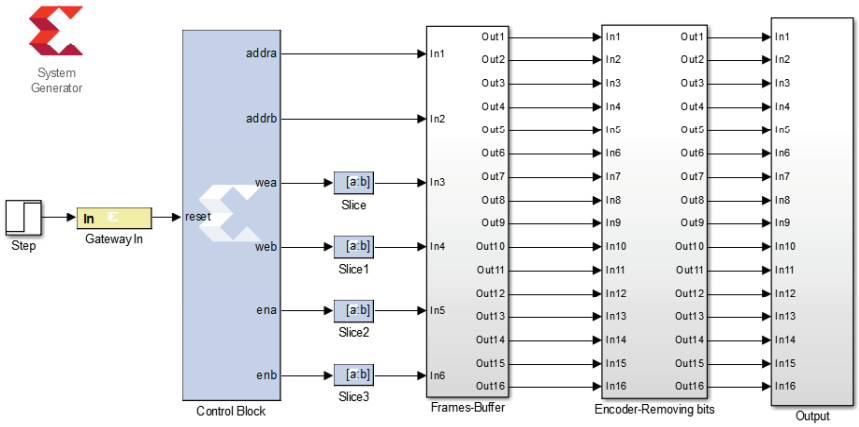

Fig. 3 Encoder architecture design

In our proposed implementation design, the image is divided into several parts of 4 different samples $x_{0,0}[n_1, n_2]$, $x_{0,1}[n_1, n_2]$, $x_{1,0}[n_1, n_2]$ and $x_{1,1}[n_1, n_2]$ such that

$$x_{p,q}[n_1, n_2] = x[2n_1 + p, 2n_2 + q]$$

with p and $q \in [0, 1]$. Then, the first one is encoded using PCM, removing l_0 LSBs, and the remaining parts using MPCM, removing l_k LSBs and m_k MSBs. A diagram of the steps used in our hardware algorithm is shown in Fig. 4. Remark that both the reading and coding of the 16 pixels is performed in the same cycle.

3.2 Decoder architecture

In Fig. 5 we show the proposed decoder architecture. In this approach, the buffer is filled with coded samples until the first three lines are completed, then the decoding process starts. The decoder is divided into two steps: decision and reconstruction. The recovery of the original image is performed as follows:

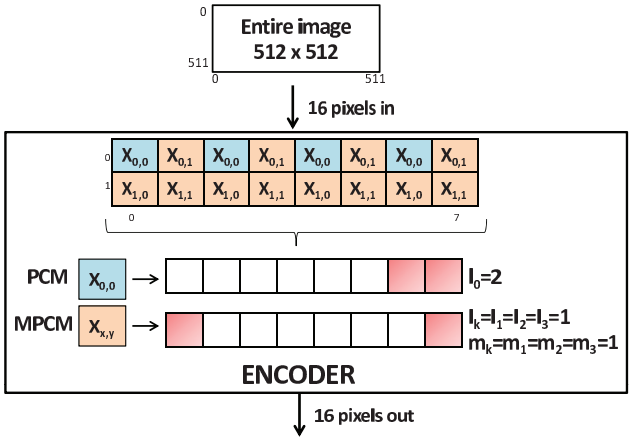

Fig. 4 Example of the encoder algorithm with $l_0 = 2$, $l_k = 1$ and $m_k = 1$

- Three columns of data are processed in decision block, where the signal PCM is reconstructed, accurate SIs are generated from PCM samples and one of the possible intervals 2^{m_k} is selected in which each MPCM decoded signal will be placed.
- At the decoder reconstruction block, we recover the LSB removed in the encoding process using its SI and interval corresponding to each MPCM sample. After completing these steps, we get 4 decoded pixels.
- In each cycle, two columns of encoded samples leave the decoder and another two enter into it. This process is carried out iteratively until all samples of the three rows from the buffer have been decoded.
- Then, the buffer is shifted. Two rows leave the buffer and two new ones enter into it. All the above operations continue until all the input encoded samples are decoded.

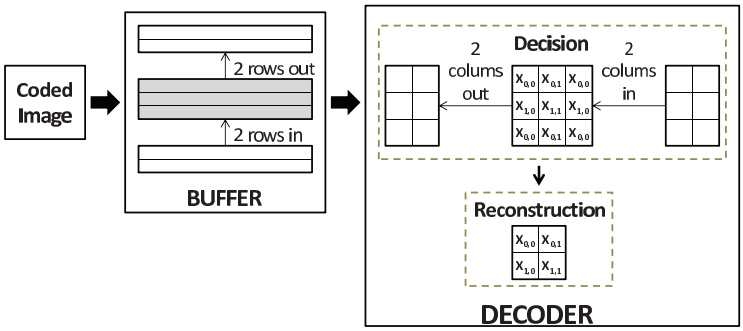

Fig. 5 Decoder architecture design

The proposed decoder design has been completely pipelined, so this makes each of the aforementioned steps used for decoding to be performed concurrently. In addition, as previously mentioned, 4 decoded pixels are obtained in each cycle. In this approach, the operating frequency has been set to get the 4 pixels, but in order to organize the entire decoded image, a Phase-Locked-Loop (PLL) module has been used, which generates multiple clocks for a given input clock. Therefore, in each cycle, 4 pixels are stored in a buffer with a fixed frequency, but these pixels are read with a frequency 4 times higher, thus achieving a serial output without delay. The buffers used in the proposed architecture have been implemented using single dual port 18 Kb block RAMs.

4 Results

Both encoder and decoder architecture designs have been tested. In this section we present the performance evaluation of the complete system in terms of PSNR, encoding/decoding times, board area usage, maximum frame rate and speed-ups obtained when compared to a CPU sequential algorithm. The architectures have been synthesized, placed and routed using Xilinx ISE 14.3 tool, and have been simulated and verified using Matlab/Simulink through System Generator toolbox. They have been designed into the Zynq AP SoC-based board previously mentioned. Occupied board area, maximum frequency and power consumption estimation have been measured from the Xilinx ISE 14.3 tool. In our experiments, we have assessed the results of five gray-scale images (*Zelda, Lena, Peppers, Barbara and Baboon*) with a resolution of 512x512 pixels and 8 bits per sample. Furthermore, we have assigned optimum values to the coding/decoding parameters so as to obtain the maximum PSNR for a given bit-rate. The assignment of encoding parameters values for the MPCM coder/decoder (with $N = 4$, $l1 = l2 = l3$ and $m1 = m2 = m3$) was proposed by Marleen Morbee in [15].

4.1 Encoder evaluation

In Table 1 the PSNR obtained for all tested images as a function of the bit-rate (R) is presented. As expected, for higher rates (R), which means removing few bits in the encoding process, MPCM algorithm generally provides good PSNR due to the fact that no significant loss occurs in the coding process, and consequently, not big errors are introduced in the decoding process. Therefore, the lower rate (R), the lower PSNR value. Note that, for each rate, parameters l_0, l_1, m_1 are not necessarily the same for all images. Each image has its appropriate parameters to get the optimal quality of the recovered image.

Regarding coding/decoding delay, the proposed encoder architecture works at a maximum clock frequency of 204,96 MHz, that is 4,879 ns. Furthermore, the algorithm requires 16387 cycles to perform the encoding process for an

	R=1bpp		R=2bpp		R=4bpp		R=6bpp	
Image	(l_0,l_1,m_1)	PSNR	(l_0,l_1,m_1)	PSNR	(l_0,l_1,m_1)	PSNR	(l_0,l_1,m_1)	PSNR
Zelda	(4,8,0)	33.83	(0,8,0)	36.57	(1,3,2)	40.07	(2,1,1)	48.07
Lena	(4,8,0)	31.79	(3,6,1)	33.71	(1,4,1)	37.74	(2,1,1)	47.80
Peppers	(4,8,0)	30.57	(3,7,0)	32.14	(4,4,0)	34.88	(4,4,0)	44.62
Barbara	(4,8,0)	24.87	(3,7,0)	26.59	(4,4,0)	33.67	(4,4,0)	44.56
Baboon	(4,8,0)	22.53	(3,7,0)	24.22	(4,4,0)	32.20	(4,4,0)	43.85

Table 1 Optimum parameters and PSNR value for all tested images

image resolution of 512x512 pixels. Therefore, we require 79,952 μs to encode any image for the aforementioned resolution, which is 12 times faster than the sequential algorithm on an Intel Core 2 CPU at 1.8 Ghz with 5 GBytes RAM. As the encoding process does not depend on the internal characteristics of the image, but only on the image resolution, in Fig. 6, the maximum frame rate achievable for the proposed architecture is presented. As shown, the hardware implementation of the MPCM encoder is able to compress up to 3558 frames per second for HD-Ready resolution or up to 1668 frames per second for Full-HD resolution.

The high speed encoding process makes high speed cameras be able to capture continuously and grab without the restrictions of the internal RAM size. For example, at a compression rate of 1 bpp, the encoding system has a throughput of 409.84 MBytes/s which is less than the available bandwidth of typical storage devices like a SSD (Solid State Drive) (up to 600 MBytes/s). Depending on the final application, if a higher image quality is necessary, the proposed MPCM hardware implementation could compress at 4 bpp rate with a good quality and a throughput bandwidth of 1640 MBytes/s which will extend the capturing time over the internal camera RAM module up to 1.4 times or will permit its transmission over an Ethernet 40 Gbit point-to-point access.

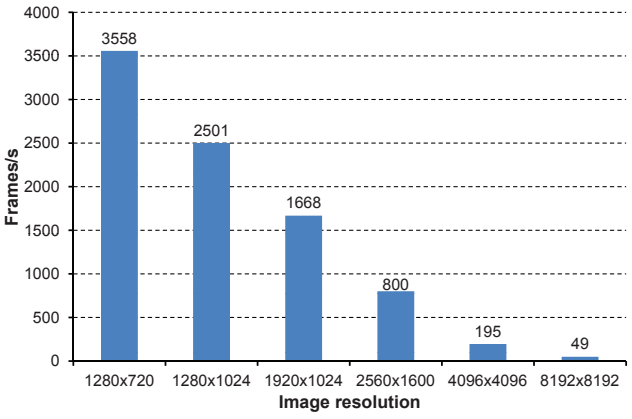

Fig. 6 Maximum encoded frames per second for different image resolutions

The basic elements of a FPGA are CLBs (Configurable Logic Blocks). CLBs architecture includes: 6-input LUTs, memory capability within the LUT and register and shift register functionality. The LUTs in the Zynq-7000 AP SoC can be configured as either one 6-input LUT (64-bit ROMs) with one output, or as two 5-input LUTs (32-bit ROMs) with separate outputs but common addresses or logic inputs. Each LUT output can optionally be registered in a flip-flop. Four such LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a configurable logic block (CLB). Four of the eight flip-flops per slice (one flip-flop per LUT) can optionally be configured as latches. Between 25-50% of all slices can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers. [20].

	Used	**Available**	**Utilization**
Number of Slices	14	13300	1%
Number of Slice Registers(as Flip Flops)	17	106400	1%
Number of Slice LUTs	35	53200	1%
Number of RAMB36	52	140	37%
FMax(MHz)	204,96	-	-
Power consumption (mW)	305	-	-

Table 2 Area used in the FPGA encoder implementation

Table 2 presents the results of the encoder implementation in terms of hardware resources used, indicating the number of used Slices, Flip-Flops, LUTs and 36 KB block RAMs. In addition, it shows an estimation of the power consumed using XPower of Xilinx ISE 14.3, being only 305 mW due to the high segmentation in the encoder design. As shown, only 1% of all the available area in the FPGA is used, so given the large amount of non used area on the FPGA, we could use it to deploy multiple identical encoders that could run concurrently. Thus, different frames could be encoded simultaneously so as to increment the available recording time of a high speed camera. To take advantage of this, we would only have to consider an external memory to support the storage of several frames, considering the blocks RAMs used as intermediate buffers.

4.2 Decoder evaluation

As far as the decoder is concerned, the maximum clock frequency has been set at 100MHz, being the lower latency 713 cycles. This frequency is taken as a compromise due to the use of other frequency 4 times higher provided by the PLL module, since analyzing the implementation delay, higher frequencies (around 160 MHz) would be achieved, as discussed in section 3.2. So, the MPCM decoder is able to recover 400 Mpixels per second at that frequency. On the other hand, the algorithm requires 66240 cycles to perform the decoding process for an image resolution of 512x512 pixels, so 662 μs are needed to

decode any image for that resolution, being 70 times faster than the sequential decoding algorithm on an Intel Core 2 CPU at 1.8 Ghz with 5 GBytes RAM.

Fig. 7 shows the maximum decoding frame rate achievable for the proposed architecture. As shown, the hardware implementation of the MPCM decoder is able to recover up to 434 frames per second for HD-Ready resolution or up to 204 frames per second for Full-HD resolution, which corresponds to a throughput of 50 MBytes/s, making available to reproduce high definition cinema at high frame rates.

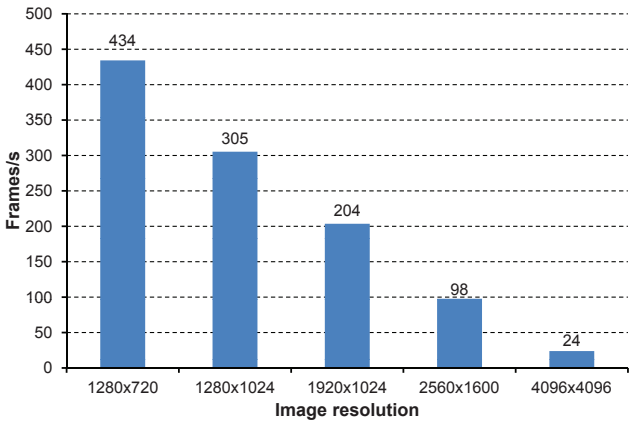

Fig. 7 Maximum decoder frames per second for different image resolutions

Regarding to occupied board area, Table 3 shows a summary of the hardware resources required by the decoder, which in a similar way than in the encoder, is less than a 1%. The occupied board area could vary depending on the l_0, l_1, m_1 parameters, but in any case it will be greater than 1%. As indicated in section 3.2, the buffers used have been modeled on single dual port 18 Kb block RAMs so as to take advantage of lower consumption compared to distributed memories, besides being faster. Note that maximum frequency shown in table 3 is 154.5 MHz, but in our design we have set this frequency to 100 MHz as explained previously.

	Used	**Available**	**Utilization**
Number of Slices	129	13300	1%
Number of Slice Registers(as Flip Flops)	415	106400	1%
Number of Slice LUTs	208	53200	1%
Number of RAMB18	5	140	4%
FMax(MHz)	154.5	-	-
Power consumption (mW)	221	-	-

Table 3 Area used in the FPGA decoder implementation for parameters $l_0 = 2$, $l_1 = 1$, $m_1 = 1$

5 Conclusions

In this paper we have presented an efficient FPGA implementation of the MPCM codec. We have show the quality of the reconstructed images in terms of PSNR at different compression rates and for several images with different textures. Regarding coding speed, the results show that our proposed implementation is able to compress a Full-HD resolution image at 1668 frames per second. The maximum achievable throughput bandwidth of our proposed implementation is 409.84 MBytes/s which permits the continuous grabbing of a nowadays high speed camera at an image resolution of HD-Ready (1280x720p) and a reasonable good quality. But, if the final application requires a higher image quality, our encoder is able to give up to 1640 MBytes/s at a 2:1 compression rate, incrementing the capturing time over the high speed camera internal RAM memory. The occupied area of the FPGA used is less than 1% of the total available area, which give us the posibility to replicate several times the encoding system and thus, several frames can be compressed in a parallel way.

We have also developed in hardware de MPCM decoder module. Our proposed decoder design is able to recover images at 204 frames per second for Full-HD resolution, whith an occupied board area of less than 1%.

References

1. J. Ahmad and M. Ebrahim. Fpga based implementation of baseline jpeg decoder. *International Journal of Electrical & Computer Sciences*, 9(9):371–377.
2. L.H. Chen, W.L. Liu, O.T.C. Chen, and R.L. Ma. A reconfigurable digital signal processor architecture for high-efficiency MPEG-4 video encoding. In *IEEE Conference on Multimedia and Expo*, 2002.
3. Xi CHEN, Lin ZENG, Qinglin ZHANG, and Wenxuan SHI. A novel parallel JPEG compression system based on FPGA. *Journal of Computational Information Systems*, 7(3):697–706, 2011.
4. A. Descampe, Devaux F., Rouvroy G., Macq B., and Legat J.D. *An Efficient FPGA Implementation of a Flexible JPEG2000 Decoder for Digital Cinema*. PhD thesis, Universit catholique de Louvain, 2002.
5. PHOTRON FASTCAM SA-X. http://www.photron.com/index.php.
6. P. Gemeiner, W. Ponweiser, P. Einramhof, , and M. Vincze. Real-time slam with high-speed cmos camera. pages 297–302, September 2007.
7. i-SPEED 3. http://www.olympus-ims.com/es/ispeed-3/.
8. O. Ismailoglu, I. Benderli, M. Korkmaz, T. Durna, Y. Kolak, and A. Tekmen. real time image processing subsystem: Gezgin. In *16th Annual/USU Conference on Small Satellites*, 2002.
9. N. S. Jayant and P. Noll. Prentice-Hall, Inc, 1974.
10. Dominique Martinez and Marc M. Van Hulle. Generalized boundary adaptation rule for minimizing rth power law distortion in high resolution quantization. *Neural Networks*, 8(6):891 – 900, 1995.
11. Vision Research PHANTOM v641. http://www.visionresearch.com/Products/High-Speed-Cameras/v641.
12. J. Prades-Nebot, A. Roca, and E. Delp. Modulo-pcm based encoding for high speed video cameras. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 153 –156, oct. 2008.
13. J. Ritter, G. Fey, and P. Molitor. Spiht implemented in a XC4000 device. In *IEEE 45th Midwest Symposium on Circuits and Systems*, 2002.

14. J. Rosenthal. *JPEG image Compression using an FPGA*. PhD thesis, University of California, 2006.
15. Marleen Morbee Ph.D. Thesis. Optimized information processing in resource-constrained vision systems, 2011.
16. Fastec Imaging TS3 100-S. http://www.fastecimaging.com/products/high-speed-cameras/handheld-cameras/ts3-100-s.
17. I. Urriza, J.I. Artigas, J.I. Garcia, L.A. Barragan, and D. Navarro. Vlsi architecture for lossless compression of medical images using discrete wavelet transform. In *Conference on Design Automation and Test in Europe*, 1998.
18. B. Vanhoof, M. Peon, and M.G. Lafruit. A scalable architecture for mpeg-4 embedded zero tree coding. In *IEEE Conference on Custom Integrated Circuits*, 1999.
19. Yan Wang, Shoushun Chen, and A. Bermak. Fpga implementation of image compression using dpcm and fbar. In *Integrated Circuits, 2007. ISIC '07. International Symposium on*, pages 329 –332, sept. 2007.
20. Xilinx Zynq-7000. Zynq-7000 all programmable soc overview, advance product specification - ds190 (v1.2) available on: http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf, August 2012.