# MEMORY REQUIREMENTS FOR HARDWARE IMPLEMENTATION OF THE H.264 ENCODER MODULES

**KAMEL.MESSAOUDI*, EL-BAY BOURENNANE*, SALAH TOUMI**, OUASSILA LABBANI***

[*] LE2I Laboratory - Burgundy University - Dijon Cedex, France
[**]LERICA Laboratory - Annaba University - Sidi Amar, Algeria
kamel.messaoudi@u-bourgogne.fr

## ABSTRACT

For a hardware implementation of any image processing algorithm, it is necessary to study the input/output of each processing module even before studying the internal architecture of these modules. And that to prepare a simulation platform, with internal and external memory, necessary to load and to prepare the input for the modules. These memories are also used as intermediate component between the different modules to provide the possibility of parallelism. In this work we give the architecture of internal and external memory used by the H.264 encoder in order to develop a simulation platform for processing modules. This platform can be realized in FPGA platform chosen according to the memory requirements.

**KEYWORDS*:*** H.264/AVC encoder, Memory management, Hardware implementation, ML501 platform.

## 1 INTRODUCTION

The H.264/AVC is the result of the collaboration between the ISO/IEC Moving Picture Experts Group and the ITU-T Video Coding Experts Group. This great video codec is the latest standard for video coding recommended for any new application.

The H.264 encoder combines several efficient algorithms, and the two groups of standarization have paid attention to the concatenation of these algorithms. Indeed, the greatest improvement of H.264 is that the decoder is used in the coding channel, in order to manipulate the same images in the encoder as in the decoder at the disadvantage of increased complexity of processing. Another improvement in H.264 is the utilization of macroblocks (MBs) with different sizes ranging from 4x4 pixels to 16x16 pixels for the luminance signal. In this work, we seek to express the memory requirements (in inputs) of each processing module of the encoder, in order to define a complete simulation platform for a hardware implementation of the H.264.

The paper is organized as follows. Section 2 presents an overview of the H.264 codec architecture. Section 3 decribes the organization of the sequence of images in a GOP and the organization of the image into a set macroblock. In section 4 a simulation platform for the H.264 encoder is given. Section 5 describes the functionality mapping of the H.264 encoder onto the external memory of images sequence and the local memories details for each precessing modules. Section 6 shows the results of simulation with ModelSim, discussions are also given. Finally, Section 7 concludes the proposal.

## 2 THE H.264 ENCODER

The new visual standard H.264 shares a number of devices with old standards [1], including the H.263 and MPEG-4. H.264 is based on a hybrid model for Adaptive Differential Pulse Code Modulation (ADPCM) and a transformation based on the coding of integers, similar to discrete cosine transform (DCT) used in earlier standards [2]. This complex coding is done to take advantage of the temporal and spatial redundancy occurring in successive visual images [3]. The diagram of the encoder H.264 is shown on the following basis [4]:

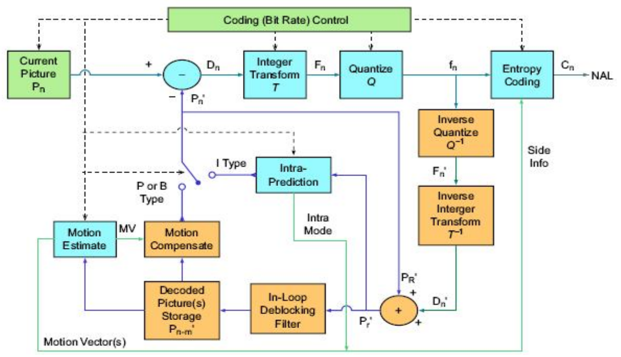


*Figure 1: H.264 encoder hardware architecture*

To show the connection of the different encoder modules with the different images of the sequence and to separate the intra-prediction mode and the inter-mode prediction, we propose the following scheme for the H.264 encoder. This diagram shows that both intra and inter modes are applied on two different images (I and P) and two different times (they must finish the intra processing to begin inter processing). In addition to this diagram shows the modules directly connected with the imput images of sequence. Subsequently in this paper we conclude that these images will be stored in external memory if we want a real-time implementation of the encoder.

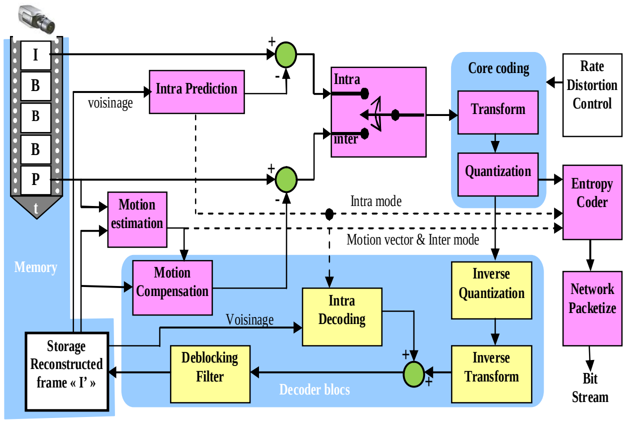


*Figure 2: Detailed H.264 encoder hardware architecture*

Given the number and the variety of applications requiring the use of the H.264 encoder, and given the number of techniques and algorithms used in series or in parallel in the encoder, the MPEG and ITU founders offer and define a set of three profiles, each supporting a particular set of coding functions. The Baseline Profile supports intra and inter-coding (using I-slices and P-slices) and entropy coding with context-adaptive variable-length codes (CAVLC). The Main Profile includes support for interlaced video, inter-coding using B-slices, inter coding using weighted prediction and entropy coding using context-based arithmetic coding (CABAC). The Extended Profile does not support interlaced video or CABAC but adds modes to enable efficient switching between coded bitstreams (SP- and SI-slices) and improved error resilience (Data Partitioning) [5][6].

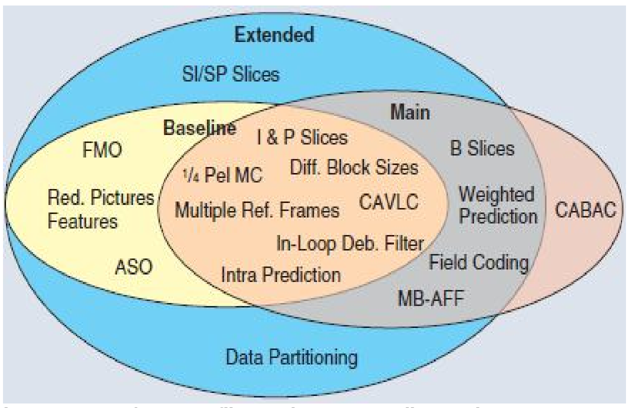


*Figure 3: H.264/AVC profiles and corresponding tools*

## 3 GOP AND MACROBLOCKS IN THE H.264 ENCODER

H.264 supports coding and decoding of 4:2:0 progressive or interlaced video. An H.264 video sequence consists of several frame types structured as GOP (Group Of Pictures). A GOP is a sequence of frames which are coded according to three methods: intra-frame coding (I-image), predictive-frame or inter-frame coding (P-image) and bidirectional-frame coding (B-image). For example, a GOP may be in the form of IBBBPBBBPBBB [7].

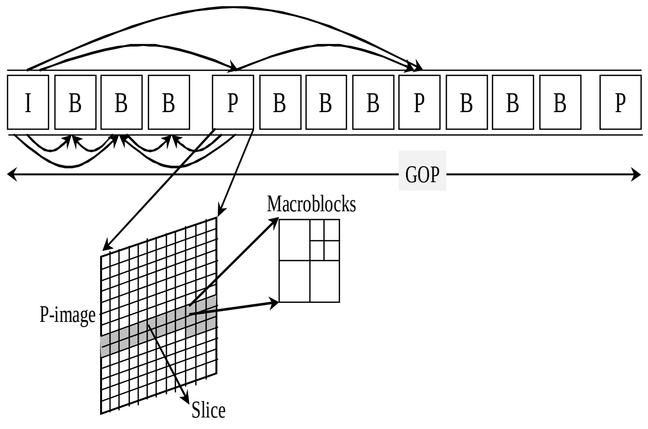


*Figure 4: The 3 frame type structured as GOP*

A coded frame consists of a number of macroblocks, each containing 16x16 luma samples and associated chroma samples (8x8 Cb and 8x8 Cr samples). I-macroblocks are predicted using intra prediction. A prediction is formed either for the complete 16x16 macroblock according to four modes, or for each 4x4 macroblock according to nine modes. P-macroblocks are predicted using inter prediction from reconstructed reference picture (I'). An inter coded macroblock may be divided into macroblock partitions, i.e. macroblocks of size 16x16, 16x8, 8x16 or 8x8 luma samples (and associated chroma samples). If the 8x8 partition size is chosen, each 8x8 sub-macroblock may be further divided into sub-macroblock partitions of size 8x8, 8x4, 4x8 or 4x4 luma samples (and associated chroma samples). Finally, B-macroblocks are predicted using inter prediction from reference frames (I' and P') [4].

## 4 SIMULATION PLATFORM FOR THE H.264 ENCODER

In figure 5, we present the principle of an implementation platform of a codec, the image is initially captured, stored in memory, processed, stored and displayed at the end (complete chain of video processing). it is also possible to register intermediate parameters and images during processing.

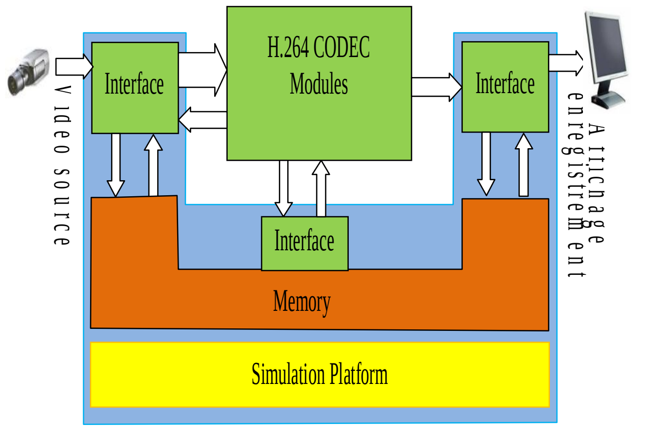


*Figure 5: Simulation platform for H.264 modules*

Processing modules in a codec are generally applied to macroblock for different sizes of recorded images, not for entire images. In addition to treating some types of images in a sequence, it is necessary to save previous images, which require us to record multiple images at once. Therefore, it is necessary to use an external memory.

## 5 MEMORY REQUIREMENT FOR EACH MODULE

Usually in a video CODEC, the local memory is used for storing the coefficients and intermediate data, and the external memory is reserved for recording images of the sequence and output files. Only for the baseline profile of the H.264 encoder with I-image and P-image only, it is possible to use local memory for saving parameters and images (depending on the uses platform prototype) [3].

With the use of external memory for storage, processing is applied independently for each GOP. The intra processing starts immediately after loading of the first picture (I-type), and inter processing starts directly after loading of the fifth picture (the first P-Frame). Then follows begin the bidirectional processing of the three B-type images. The same for other images of the same GOP:

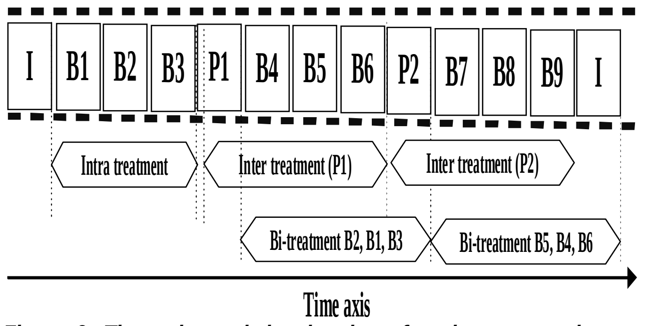


*Figure 6: The order and the duration of each treatment between different images in a GOP*

Note that different processing is applied to the MBs of different sizes (not to the full images). In the flow diagram, we present the modules in direct contact with the external memory and their memory requirements acording to the image type and the processing mode.

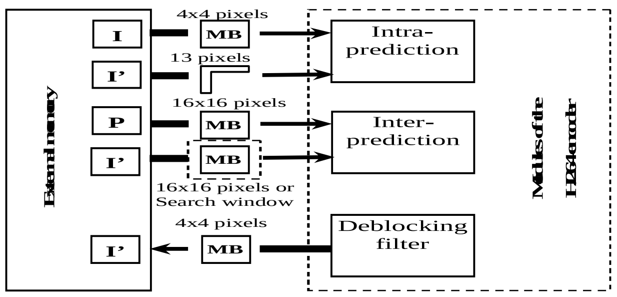


*Figure 7: Modules of the H.264 encoder in direct contact with the external memory*

For other modules of the H.264 encoder, we propose the use of other local memories between each two processing modules in order to have a pipelined architecture. Generally these memories have a 4x4 pixels size. Infact, the modules of the encoder (DCT/IDCT, Q/IQ, deblocking filter and the entropy coding) process MBs of 4x4 pixels size:

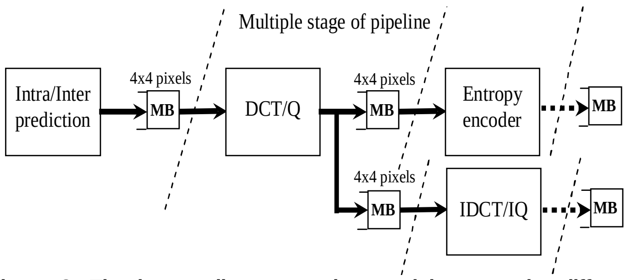
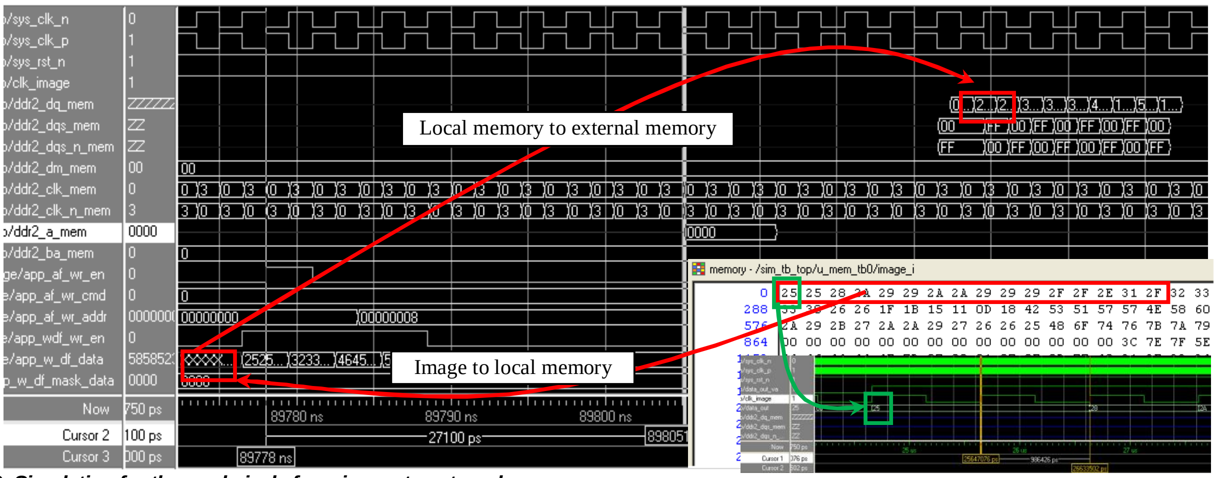


*Figure 8: The intermediate memories used between the different modules of the H.264 encoder*

For this architecture, we can calculate the percentage of occupency of internal and external memory used by the H.264 encoder. This memory design step allows us to choose the FPGA and to build the prototype platform.

## 6 SIMULATIONS RESULTS

The aim of this work is to calculate and to implement the various internal and external memories required for an hardware implementation of the H.264 encoder, and to find a strategy for loading the various internal memory from external memory that contain the images of sequence. In simulation, a test bench file is written to replace the image capture part, and the file must provide the image pixels at a frequency (fpixels) calculated according to the type of input video sequence format CIF with 352x288 pixel/image and 10 images/Second, the pixel frequency is given by 352x288x10Hz. The following figure shows the method for reading pixels from the image file to the external memory:



```
  0  25 25 28 2A 29 29 2A 2A 29 29 29 2F 2F 2E 31 2F 32 33
288  33 30 26 26 1F 1B 15 11 0D 18 42 53 51 57 57 4E 58 60
576  2A 29 2B 27 2A 2A 29 27 26 26 25 48 6F 74 76 7B 7A 79
864  00 00 00 00 00 00 00 00 00 00 00 00 00 00 3C 7E 7F 5E
```

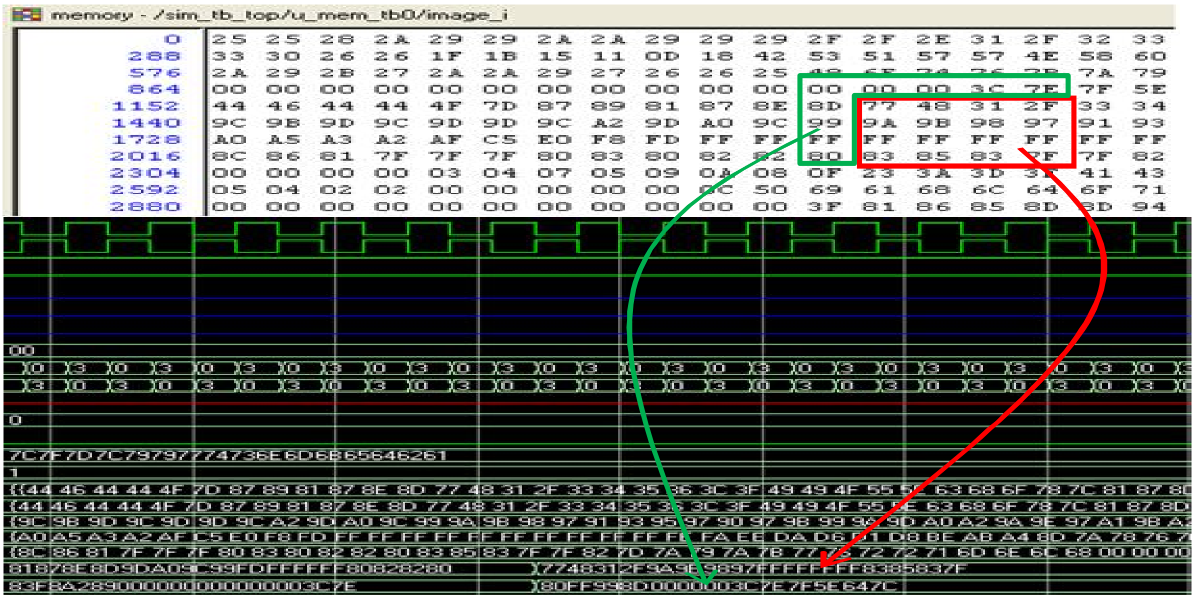
*Figure 9: Simulation for the read pixels from image to external memory*

Figure10 shows a reading of 4 lines of an image from external memory and the recording of these lines in an internal memory (mem_4row).

Reading from external memory to local memory is performed periodically as required H.264 encoder modules. For example, the intra4x4 module of the encoder is needed for each intra prediction of a 4x4 pixels macroblock and 13 pixels neighborhood. The 4x4 macroblock is read directly from the external memory to the local memory and the neighborhood pixels are reused from previous calculations. So at the end of each calculation of a 4x4 MB, it saves the pixels that are as near to other MBs. The figure 12 shows the reading of a 4x4 MB and the designation of neighboring pixels.

Similarly for other modes of intra prediction and inter-prediction. So our proposal method that supplies periodically modules of the H.264 encoder by macroblocks from the sequence of prerecorded images at the local memory.

The calculation of memory allows us to choose the necessary FPGA platform for implementation of the encoder. In our case we choose the Virtex5 ML501 platform from Xilinx (available at the laboratory LE2I) which appear rich in memory and programmable logic resources to satisfy a hardware implementation of H.264 encoder. The table1 shows the percentage use of memory resources for two types of image sequence:



```
   0  25 25 28 2A 29 29 2A 2A 29 29 29 2F 2F 2E 31 2F 32 33
 288  33 30 26 26 1F 1B 15 11 0D 18 42 53 51 57 57 4E 58 60
 576  2A 29 2B 27 2A 2A 29 27 26 26 25 48 6F 74 76 7B 7A 79
 864  00 00 00 00 00 00 00 00 00 00 00 00 00 00 3C 7E 7F 5E
1152  44 46 44 44 4F 7D 87 89 81 87 8E 8D 77 48 31 2F 33 34
1440  9C 9B 9D 9C 9D 9D 9C A2 9D A0 9C 99 9A 9B 98 97 91 93
1728  A0 A5 A3 A2 AF C5 E0 F8 FD FF FF FF FF FF FF FF FF FF
2016  8C 86 81 7F 7F 7F 80 83 80 82 82 80 83 85 83 7F 7F 82
2304  00 00 00 00 03 04 07 05 09 0A 08 0F 23 3A 3D 3F 41 43
2592  05 04 02 02 00 00 00 00 00 0C 50 69 61 68 6C 64 6F 71
2880  00 00 00 00 00 00 00 00 00 00 00 3F 81 86 85 8D 8D 94
```

```
00
0
7C7F7D7C79797774736E6D6B65646261
1
{{44 46 44 44 4F 7D 87 89 81 87 8E 8D 77 48 31 2F 33 34 35 36 3C 3F 49 49 4F 55 5E 63 68 6F 78 7C 81 87 8D
{44 46 44 44 4F 7D 87 89 81 87 8E 8D 77 48 31 2F 33 34 35 36 3C 3F 49 49 4F 55 5E 63 68 6F 78 7C 81 87 8D
{9C 9B 9D 9C 9D 9D 9C A2 9D A0 9C 99 9A 9B 98 97 91 93 95 97 90 97 9B 99 9A 9D A0 A2 9A 9E 97 A1 9B A
{A0 A5 A3 A2 AF C5 E0 F8 FD FF FF FF FF FF FF FF FF FF FF FA EE DA D6 D1 D8 BE AB A4 8D 7A 78 76 7
{8C 86 81 7F 7F 7F 80 83 80 82 82 80 83 85 83 7F 7F 82 7D 7A 79 7A 7B 77 75 72 72 71 6D 6E 6C 68 00 00 00
81878E8D9DA09C99FDFFFFFF80828280        }77483 12F9A9B9897FFFFFFFFF8385837F
83F8A28900000000000000003C7E            }80FF998D00000003C7E7F5E647C
```

*Figure 11: Read memory for the 4x4 pixels intra-prediction*

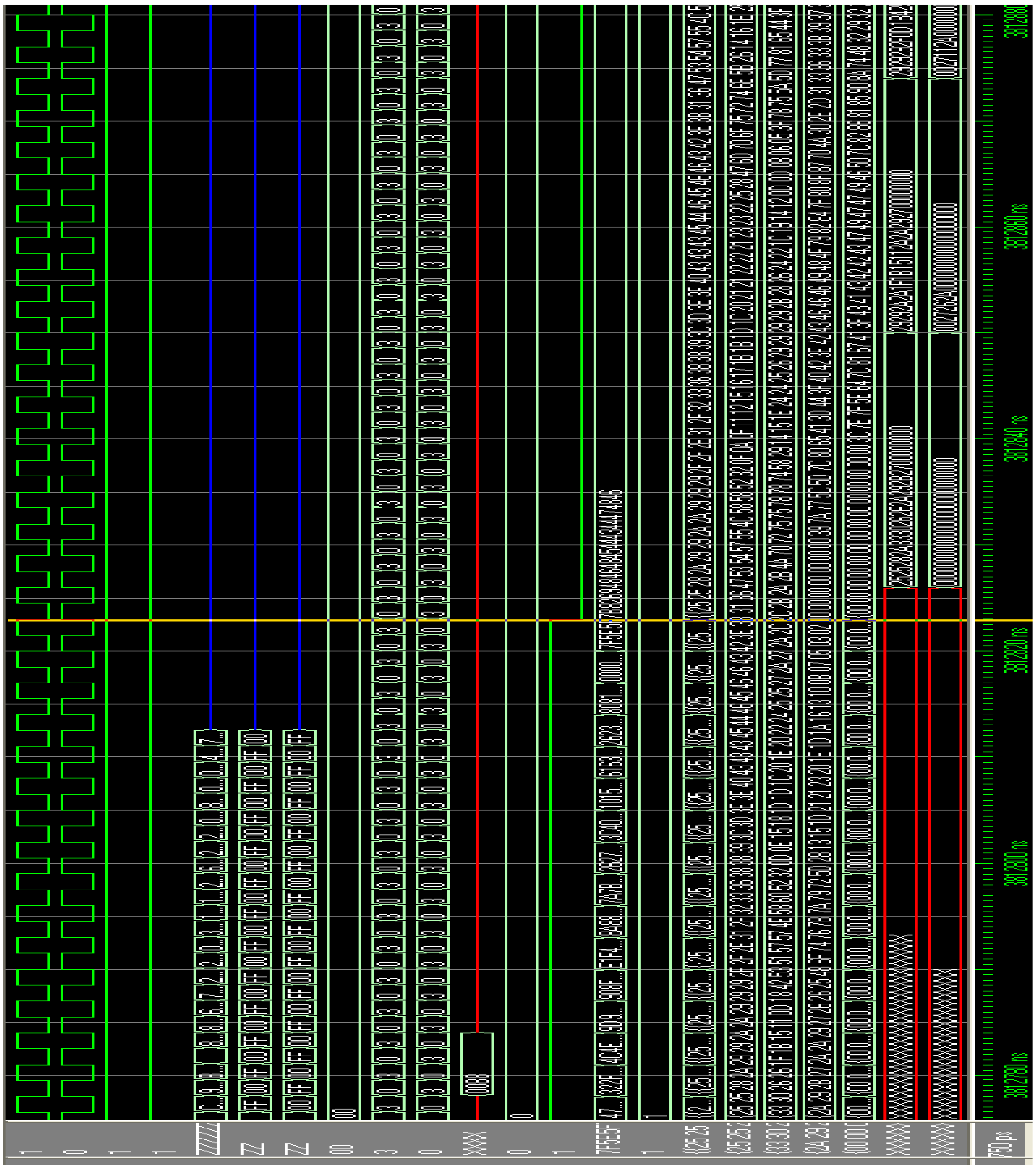


***Figure 10: Reading of 4 lines of an image from external memory***

***TABLE 1: The external and local memory percentage***

| **Profile** | | **External memory used** | | **Local memory used** | |
|---|---|---|---|---|---|
| | | *CIF Image* | *HDTV Image* | *CIF Image* | *HDTV Image* |
| **Baseline** | *Capacity* | 297 KB | 2,637 MB | 101,875 Kb | 240,25 Kb |
| | *percentage* | 0,113% | 1,03% | 5,89 % | 13,90 % |
| **Main** | *Capacity* | 1,16 MB | 10,747 MB | 101,875 Kb | 240,25 Kb |
| | *percentage* | 0,453% | 4,12% | 5,89 % | 13,90 % |

## 7 CONCLUSION

In this work we have shown the needs of memories for different the H.264 encoder modules with the goal of a hardware implementation of these modules in a real simulation platform, estimation of the memory space (internal and external) necessary for this implementation has been calculated for two types of image sequences CIF (352x288 pixels - 10 fps) and HDTV (1280x720 - 30 frames / sec). The maximum memory used on the ML501 platform Virtex5 Xilinx, is 13.90% which makes it possible to use this platform for a hardware implementation of H.264 encoder. Simulation results concerning the filling of internal memories from an external memory of the platform are also given.

## REFERENCES

**[1]** ISO/IEC 14496–10:2003, "Coding of Audiovisual Objects-Part 10: Advanced Video Coding," 2003, also ITU-T Recommendation H.264 "Advanced video coding for generic audiovisual services".

**[2]** K. Babionitakis, G. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, I. Sifnaios, and N. Vlassopoulos, "A real-time H.264/AVC VLSI encoder architecture," Springer, Real-Time Image Proc, pp.43–59, 2009.

**[3]** K. Messaoudi, S. Toumi, E. Bourennane, "Material architecture proposition for the block matching method of motion estimate in H264 standard," ICTTA'08, Damascus, Syria, Jul. 2008.

**[4]** K. Messaoudi, S. Toumi, E. Bourennane, "Proposal and study for an architecture hardware/software for the implementation of the standard H264," CISA'08, Mediterranean Conference on Intelligent Systems and Automation- Proceeding editor AIP (American Institute of Physics), Annaba-Algeria, jun. 2008.

**[5]** J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, Performance, and Complexity," IEEE, Circuits and Systems Magazine, pp. 7-28, First Quarter 2004.

**[6]** T. Chen, C. Lian and L. Chen, "Hardware Architecture Design of an H.264/AVC Video Codec," IEEE – 7D-3, pp. 750-757, 2006.

**[7]** I. E. G. Richardson, "H.264 and MPEG-4 Video Compression," 2003, The Robert Gordon University, Aberdeen, UK, WILEY edition 2003.