A Real-Time Simulation Platform for the H.264 CODEC Modules

Kamel Messaoudi[12] Maamar Touiza[1] ElBay Bourennane[1] Salah Toumi[2] Ouassila Labbani[1]

[1] *LE2I Laboratory - Burgundy University - Dijon Cedex, France*
(kamel.messaoudi@u-bourgogne.fr) , (maamar.touiza@u-bourgogne.fr)
(ebourenn@u-bourgogne.fr) , (ouassila.labbani@u-bourgogne.fr)
[2] *LERICA Laboratory - Annaba University - Sidi Amar, B.P# 12, Annaba, Algeria*
(toumi.salah@hotmail.com)

Abstract — ***In this paper, we propose a real-time platform for H.264 CODEC with a memory management method in which we use a preloading mechanism in order to reduce external memory accessing. The simulation platform uses then an external DDR2 memory to record the images of the sequence and an intelligent memory controller which reads the DDR2 periodically to charge another local memory for the processing modules of the H.264 encoder containing macroblocks (of different size). We either use image manipulation or chosen mode. The proposed intelligent controller is tested on an Xilinx-virtex5-ML501 platform with multiple internal and external components, including a DDR2 of 256MB. Similarly, the proposed memory controller unit is well-adapted to future memory-bandwidth-constraint System-on-Chip applications.***

Keywords: Memory management, H.264/AVC CODEC, ML501 platform, Real-time application.

I. INTRODUCTION

The H.264 video coding standard has achieved a significant improvement in coding efficiency when compared to other coding methods. However, the computational complexity of the H.264 encoder is increased drastically, resulting on practical difficulties in its implementation on the embedded platform. To use this standard method in real-time applications, it is necessary to implement on hardware accelerators.

A. The H.264 Standard

The new video coding standard recommendation H.264 of ITU-T (also known as international standard 14496-10 or MPEG-4 part 10 Advanced Video Coding 'AVC' of ISO/IEC [1]) is the latest coding standard in a sequence of the video coding standards H.261 (1990) , MPEG-1 Video (1993), MPEG-2 Video (1994), H.263(1995, 1997), MPEG-4 Visual or part 2 (1998) [2].

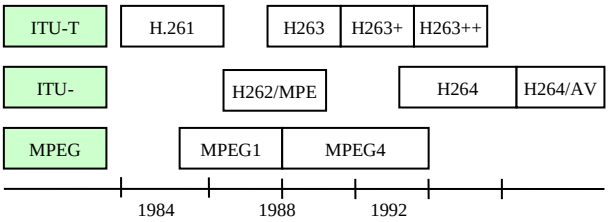


Fig. 1. Video coding standards evolution.

These previous coding standards reflect the technological progress in video compression and the adaptation of video coding to different applications and networks. Applications range from video telephony (H.261) to the internet or mobile networks (H.263, MPEG-4). The importance of new network access technologies demand for the new video coding standard H.264/AVC, providing enhanced video compression performance in view of interactive applications such as video telephony that requires a low latency system and non-interactive applications for the storage, broadcast, and the streaming of standard definition TV where the focus is on high coding efficiency [2].

TABLE I. Applications for standard video coding.

Standard	Application
MPEG-1	Video-CD, CDI
MPEG-2	Digital Television, High definition TV, DVD, TNT, Satellite video
MPEG-4 Visual	Internet, Mobil video, Studio, Video On Demand (VOD)
MPEG_4 AVC H.264 AVC	Internet, Mobil video, Studio, High Definition Video On Demand

Comparing the H.264/AVC video coding tools (multiple reference frames, 1/4 pel motion compensation, deblocking filter or integer transform) to the tools of previous standards video coding, the H.264/AVC brought in the most algorithmic discontinuities in the evolution of standardized video coding[3].

B. Hardware Implementation of the H.264 CODEC

In the literature, there are many hardware architectures proposed for the H.264 modules [4] [5]. There is a plethora of results related to the design and realization of modules of a H.264/AVC encoder [4-6-7-8], but there are only a few papers published in the literature that are related to the simulation platform and to the memory part that provides the entries to the modules. Such entries are needed to obtain complete hardware H.264 encoder solutions.

C. Paper organization

The paper is organized with Sect. 2 includes which presents an overview of the H.264 CODEC architecture, the mapping of the functionality of the H.264 encoder onto the memory of images sequence and the details of the modules in direct contact with the memory. Section 3 describes the proposal architecture for the simulation platform for the H.264 modules at the base of DDR2 memory. Section 4 shows the results of simulation with ModelSim, discussions are also given. Finally, Sect. 5 concludes the proposal.

II. The H.264 Encoder Structure

The new visual standard H.264 shares a number of devices with old standards, including H.263 and MPEG-4. Mainly, H.264 is based on a hybrid model for the 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. 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 H264 is shown on the following basis:

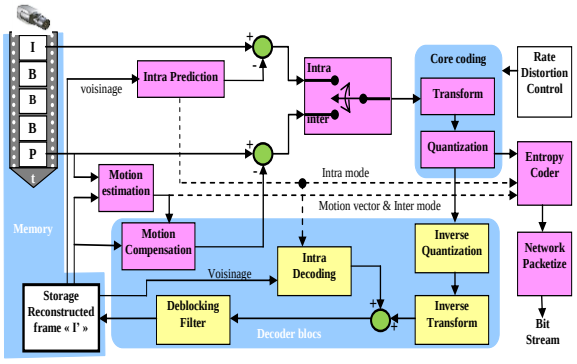


Fig. 2. H.264 encoder hardware architecture.

In this figure, only two modules (intra prediction and inter prediction) are in direct contact with the memory (mode read of image sequence), a detailed study of these two modules about reading blocks and macroblocks is given in what follows.

A. H.264 Modules Description

The H.264 encoder, when taken as a system, processes video frames that are divided in basic units defined as MacroBlocks (MBs). Each MB is a square tile of 16x16 luminance and 8x8 chrominance data. The entire encoding operation is composed of the forward (encoding) path and the inverse (reconstruction or decoding) path. The forward-encoding path predicts each MB using Inter or Intra prediction and also transforms and quantizes (TQ) the residual, and then it forwards the result to the Entropy Encoder module and finally forms the output packets in the Network Abstraction Layer (NAL) module. The inverse path involves the reconstruction of the MB from the previously transformed data by utilizing the Inverse Transform and Quantization (ITQ) and the deblocking filter [4].

B. Profiles and Levels

H.264 defines 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) [2].

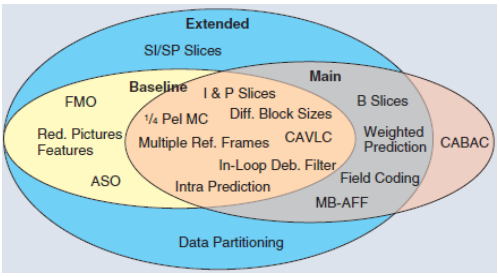


Fig. 3. H.264/AVC profiles and corresponding tools.

C. Video Format in the H.264 encoder

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

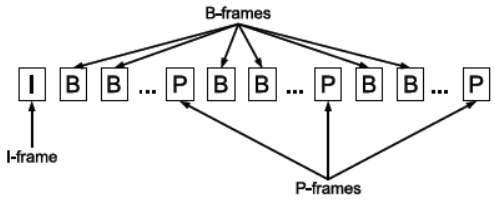


Fig. 4. The 3 frame type structured as GOP.

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') [3].

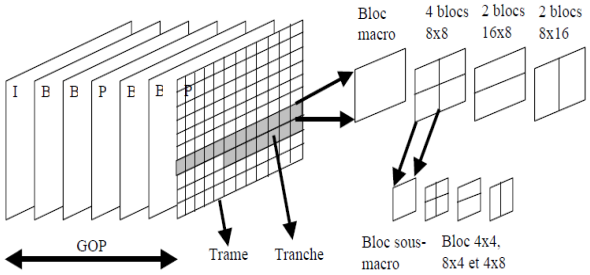


Fig. 5. Frame decomposition in H.264.

D. Intra Prediction

In intra mode a prediction macroblock (Ip) is formed based on previously encoded and reconstructed macroblocks and is subtracted from the current block prior to encoding. For the luma samples, (Ip) is formed for each 4x4 macroblock or for a 16x16 macroblock. There are a total of nine optional prediction modes for each 4x4 luma macroblock, four modes for each 16x16 luma macroblock and four modes for the chroma components. The encoder typically selects the prediction mode for each macroblock that minimises the difference between (Ip) and the macroblock to be encoded in (I) [4].

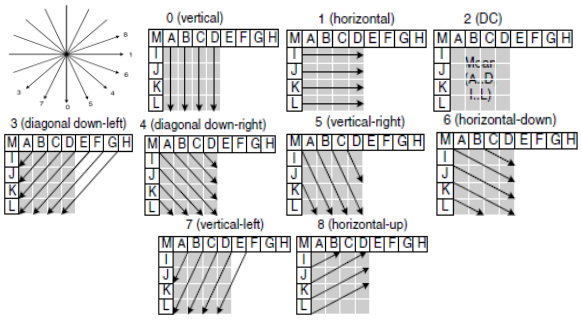


Fig. 6. Intra 4x4 prediction modes.

E. Inter Prediction

In former standards as MPEG-4 or H.263, only macroblocks of the size 16x16 and 8x8 are supported. A displacement vector is estimated and transmitted for each macroblock, refers to the corresponding position of its image signal in an already transmitted reference image. Important differences from H.264 standards include the support for a range of block sizes (from 16x16 down to 4x4) and fine subsample motion vectors (quarter-sample resolution in the luma component) [5].

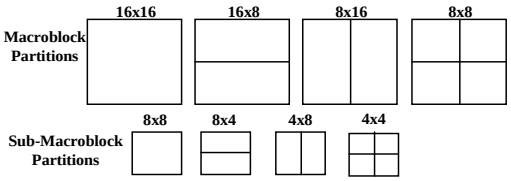


Fig. 7. Partitioning of a macroblock and a sub-macroblock.

These partitions and the sub-macroblock give rise to a large number of possible combinations within each macroblock. This method of partitioning macroblocks into motion compensated sub-blocks of varying size is known as tree structured motion compensation. A separate motion vector is required for each partition or sub-macroblock. Each motion vector must be coded and transmitted and the choice of partition(s) must be encoded in the compressed bitstream. Choosing a large partition size means that a small number of bits are required to signal the choice of motion vector(s) and the type of partition but the motion compensated residual may contain a significant amount of energy in frame areas with high details. Choosing a small partition size may give a lower-energy residual after motion compensation but requires a larger number of bits to signal the motion vectors and the choice of partition(s). The choice of partition size has then a significant impact on compression performance. In general, a large partition size is appropriate for homogeneous areas of the frame and a small partition size may be beneficial for detailed areas [4].

F. The H.264 Decoder

The decoding process consists of interpreting the coded symbols of a compliant Bitstream and of processing this data according to the standard specification in order to generate a reconstructed video sequence. Thus, the decoding process consists generally of two primary paths: the generation of the predicted video macroblocks and the decoding of the coded residual macroblocks. The sample values of the output macroblocks resulting from these two paths are summed and clipped to the valid range for pixel data to form the reconstructed macroblocks. A generalized block diagram of an H.264/AVC decoder is given in the following Figure [9]:

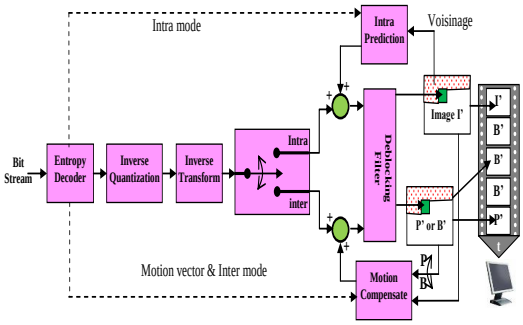


Fig. 8. H.264 Decoder hardware architecture.

G. The H.264 and Real-Time Applications

Nowadays, many video applications require the H.264 encoder to perform at real-time with low bit-rate. Moreover, these applications generally require a hardware implementation with minimal VLSI cost. The problem of optimizing both the bit-rate (compression ratio) and the VLSI area introduces new challenges in the field of hardwired video encoders.

1. The choice of a profile which has to meet an acceptable bit-rate by providing performance close to that of the reference H.264 encoder.
2. The design of the flow graph which has to minimize the internal memory and the register count.
3. Providing an efficient mapping of the computations onto the hardware resources has to maximize the utilization of the computing resources and to allow the elimination of those resources that show small utilization rates.
4. The designers have to implement an architecture with low clock rates and minimized power consumption.

To meet these challenges, the designer has to look for a set of features of the chosen profile, resulting in the required performance with respect to the compression ratio and to the area architecture.

III. Simulation Platform for the H.264 Codec

To meet the real time requirements of a large number of multimedia applications, researchers and engineers have presented solutions for realizing the CODEC or for optimizing the various modules of the encoder/decoder, but the memory part that provides the input values to the encoder modules is generally not described. In-fact, most solutions assume that the entries of encoder's modules are a bus (4, 8, 16, 32, ...) of bits or pixels, but the recording and management of these pixels are not given, or are supposed insured by software.

In the flowing figure, 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 during processing, these parameters can even be an entire image.

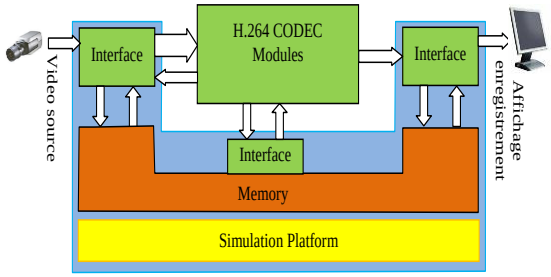


Fig. 9. Simulation platform for H.264 modules.

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.

A. Local and External Memory

In simulation platforme, the local memory organization is of significant importance because it provides a temporal storage and the means for efficient communication of the aforementioned data among the CODEC modules. Usually in codec, the local memory is used for recording the coefficients and intermediate data, and the external memory is reserved for recording images of the sequence and output files. Only for the basic profile of CODEC with I-image and P-image, that it is possible on any platform prototype, to use local memory for saving parameters and images [3].

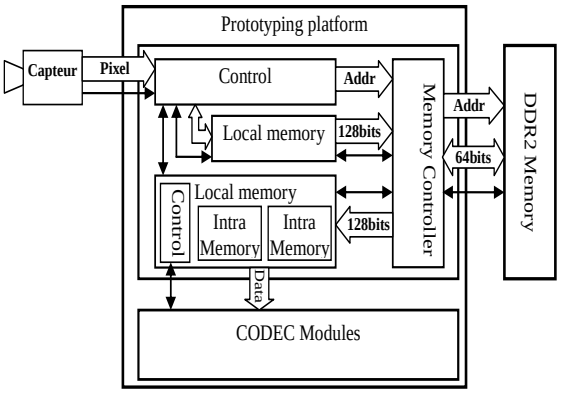


Fig. 10. Proposed memory management for H.264 modules.

B. Memory Management

In the previous figure, the images of sequence are stored in external DDR2 memory, after gathering packet of 128 bits in a local memory. Next, the DDR2 memory has fueled another local memory periodically by the macroblock processing object following the mode of treatment and the type of image.

For intra16x16 mode, the local memory must contain a macroblock (16x16 pixels of the image 'I') and the 33 neighboring pixels. For intra4x4 mode, the memory must contain the 4x4 pixels and the 13 surrounding pixels (as shown in the figure 6). In inter mode, memory must contain the macroblock of 16x16 pixels image 'P' and a macroblock of 16x16 pixels image 'I''. This last macroblock must be recharged periodically according to a scan mode from a search window in the reference image 'I'', to achieve this architecture two proposals are appropriate:

- Download the search window in one goes that from the external memory to local memory, with the disadvantage of the need of a local memory.

Use only 16x16 pixels memory, and charge this for each calculation of SAD, with the disadvantage of the large number of memory access (for eatch macrobloc 16x16) and the delay time for each access. To resolve this problem, we can use two memories (16x16 pixels) instead of a single-mode ping-pong, with a memory load step and another step

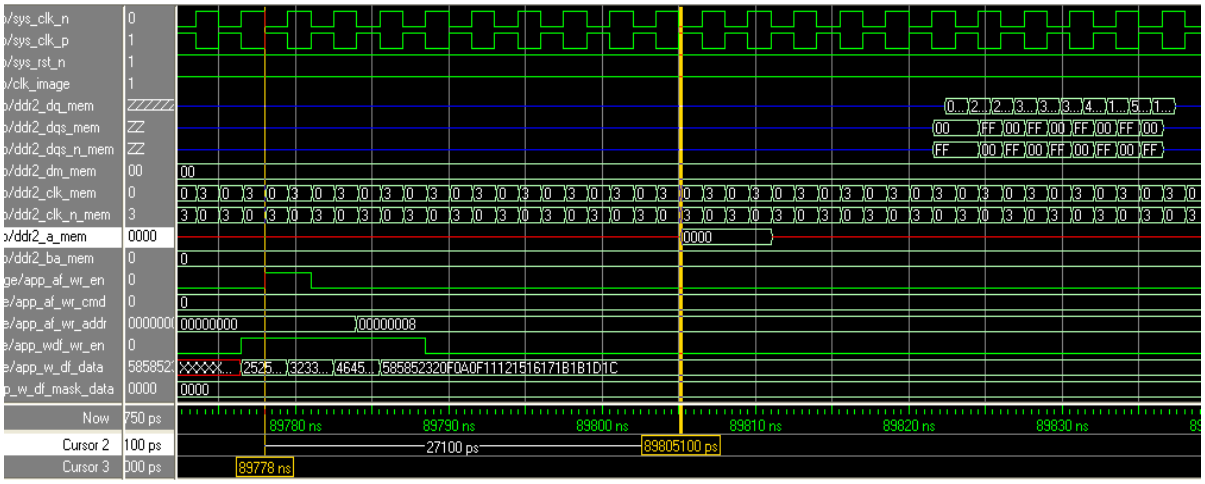
- in literacy and numeracy. The ping-pong memory mostly doubles the memory requirements.

C. Prototyping Platform

The prototyping was done using the Xilinx Virtex5 ML501 development platform, with many external components (serial port, VGA output, many different inputs, and other interface resources with de FPGA core type XC5VLX50-1FFG676). It was also fitted with a DDR2 SDRAM SODIMM module of 256MB.

The FPGA contains multiple IP blocks: RAMs 36 Kbit blocks configured according to application needs, FIFO blocks, programmable logic blocks, DSPs, clock generators, ... [10].

D. DDR2 Memory Organization



DDR2 memory is organized into 4 banks, each bank is in the form of a matrix of cells with 210 columns and 213 rows, each cell is composed of 64bits. For the luminance signal Y, each cell may contain, therefore, 8 pixels. Similarly, a line can contain 8x210 pixels, and an image 256x256pixels has 8 lines of DDR2 memory [11].

From this architecture of DDR2 memory, that it is impossible to read an amount less than 8 pixels. This fact, reads image block for the H.264 standard, performed by 8, 16, 32, ... pixels at a time. In addition to the mode (Burst = 4), we can read (4x128bits), which means 32 pixels at a time.

Reading from DDR2 to local memory is performed periodically as required H.264 modules CODEC. For example, as mentioned in paragraph 3.2, the module intra4x4 of the encoder is needed for each intra prediction of a 4x4 pixels that are as near to other blocks. The figure below shows the reading of a 4x4 block and the designation of neighboring pixels.

IV. Simulation Results

At simulation, a test bench file is written to replace the image capture part, the file must provide the image pixels at a frequency (fpixels) calculated

according to the type of input video sequence (a sequence format CIF with 352x288 pixel/image and 10 images/Second, the pixel frequency is given by 352x288x10Hz).

At the input of the Controller, the pixels are collected by a group of 16 pixels by 4 times (128 bits x 4 => Burt mode) at frequency fpixels, then transfer the 64 pixels (512bit) to the DDR2 memory through the DDR controller at the frequency of FPGA (100 MHz for the platform ML501). The following figure shows the timing of the transfer of the first pixels of the image to DDR2 memory:

macroblock and 13 pixels neighborhood. The 4x4 macroblock is read directly from the DDR2 to the local memory and the neighborhood pixels are reused from previous calculations. So at the end of each calculation of a 4x4 block, it saves the pixels that are as near to other blocks. The figure below shows the reading of a 4x4 block and the designation of neighboring pixels.

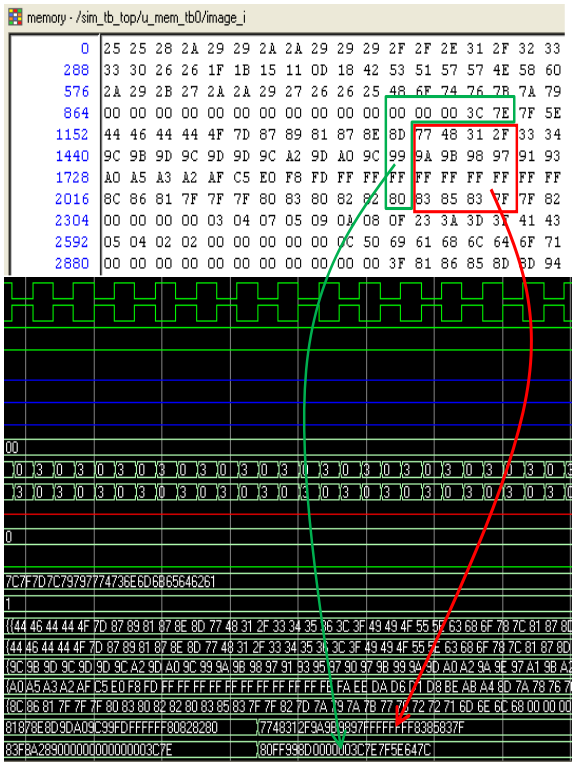


Fig. 12. Read memory for the 4x4 pixels intra-prediction.

Similarly for other modes of intra prediction and inter-prediction. So our proposal is an intelligent controller that supplies periodically modules of the H.264 encoder by macroblocks from the sequence of prerecorded images at the DDR2 in the ML501 platform. The modules of the encoder deal therefore only with part of the calculation, not with the reading part of macroblocks.

V. Conclusion

In this work, we optimize an intelligent memory controller for the H.264 encoder to build a real-time platform for simulation with the CODEC modules. A series of optimization is applied to the controller, to enable him to supply the modules of treatement periodically by the required data (macroblocks) for each step. Simulations were performed with ModelSim using as Virtex5 ML501 platform. This study allowed us also to set the time (the number of clock cycles) required for processing each macroblock of different images and that following the types of images manipulated. The proposed platforme with intelligent controller can be employed in many real-time implementation of the H.264 CODEC with advantage of high flexibility and reconfiguration.

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] 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”, 2004, IEEE Circuits and Systems Magazine, pp. 7-28, First Quarter 2004,

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

[4] 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”, 2008, Springer - Real-Time Image Proc, pp.43–59.

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

[6] D. T. Lin and C. Yang. Wu, “H.264/AVC Video Encoder Realization and Acceleration on TI DM642 DSP”, 2009, Springer - Springer-Verlag Berlin Heidelberg 2009, pp. 910-920.

[7] T. Liu, T. Lin, S. Wang, W. Lee, K. Hou, J.Yang and C. Lee, “An 865-µW H.264/AVC Video Decoder for Mobile Applications” 2005, IEEE – 12-2, pp. 301-305.

[8] S. Wang, S. Yang, H. Chen, C. Yang and J. Wu, “A Multi-core Architecture Based Parallel Framework for H.264/AVC Deblocking Filters”, 2008, Springer - Sign Process Syst.

[9] M. Horowitz, A. Joch, F. Kossentini, S. Member, and A. Hallapuro, “H.264/AVC Baseline Profile Decoder Complexity Analysis”, 2003, IEEE transactions on circuits and systems for video technology, vol. 13, no. 7, pp.704–716, july 2003.

[10] Xilinx DOC, « Xilinx Memory Interface Generator (MIG) User Guide, DDR SDRAM, DDRII SRAM, DDR2 SDRAM, QDRII SRAM and RLDRAM II Interfaces », UG086 (v2.0) September 18, 2007.

[11] Micron Technology, Inc, « 128MB, 256MB, 512MB: (x64, SR) 200-Pin DDR2 SDRAM SODIMM Features », pdf: 09005aef80eec96e, 2004, 2005 Micron Technology.