DEVELOPMENT OF A SOFTWARE MODEL FOR AN H.264/AVC PROGRESSIVE MAIN PROFILE HARDWARE VIDEO DECODER

Alonso A. Schmidt, Fábio F. Vidor, Márlon A. Lorencetti, Altamiro A. Susin

Universidade Federal do Rio Grande do Sul

ABSTRACT

This work presents a model software for the development of an H.264/AVC hardware decoder. The software is capable of decoding progressive video bitstreams up to the main profile. Also, it is able to generate test vectors to aid the validation of the FPGA prototypes. Furthermore, the software was employed for study of the complex video coding standard. During the development of this software, the group could learn about simple ways of implementing the H.264/AVC features, which could help the design of efficient hardware solutions, since the software allows the designer to know the interfaces and performance requirements. Some tests were performed to check for possible optimizations. The modular structure of the software, resembling a hardware implementation, has been proved more practical than the use of JM software for validation purposes.

1. INTRODUCTION

This work is part of the Rede H.264 project [1] for the development of the SBTVD (Sistema Brasileiro de Televisão Digital) [2], which has as main objective to develop an FPGA prototype of an H.264/AVC video decoder that will be later implemented into ASIC technology. It presents the continued development of a reference software [3,4] for validation and modeling of an H.264/AVC video decoder.

Aiming higher compression, the H.264 standard was developed, including several novelties such as adaptive entropy coding and intra frame prediction, which presents greater computational complexity. Such complexity makes the development of hardware more suitable to be done in an incremental approach, where each module is designed separately, concerning only about previously defined interface signals for later integration in the whole decoder. To support the hardware development, a modeling software, called PRH.264 [3], was created to acquire knowledge about the algorithms involved in the decoding process and which interface and signals would be necessary in the development. This program is also able to extract intermediary data from the decoding process which can be used as test vectors to later validate the hardware.

The software development has begun from a source code with the most essential features of the H.264/AVC standard, covering only a subset of the baseline profile, without in-loop deblocking filter, multiple reference pictures for motion compensation, and multiple slices support. These missing features were gradually implemented into the software until it was capable of decoding progressive video coded with main profile features.

The techniques used for achieving high compression in the main profile are presented in the section 2. In section 3, the algorithms written to implement these features are broadly discussed, as well as the validation with the JM Reference Software [5]. Section 4 presents dynamic program analysis made with different video bitstreams generated by JM software and the identification of the processes with the higher computational costs. In section 5, it is presented what could be learned with the model and the results.

2. H.264/AVC MAIN PROFILE FEATURES

In the H.264/AVC standard, images (frames or fields) are separated into one or more slices. Each slice is divided into 16x16 pixel samples regions called macroblocks, which are decoded one by one. Pixel samples can be represented as a composition of prediction and residual information. For the process of macroblock decoding, some information regarding its prediction mode is acquired by the entropy decoder and supplied to one of the predictors block. There are two types of prediction: spatial and temporal. The prediction modes can be denoted by macroblock types: I, P or B. I macroblocks use intra (spatial) prediction; P and B macroblocks uses inter (temporal) prediction. In a P slice, there can be P-type or I-type macroblocks only. In a B slice, there can be B-type or I-type only macroblocks. In an I slice there can be only I-type macroblocks. A residual that first passes through a process of inverse quantization and inverse transform is added to the macroblock predicted samples. After the whole image has been decoded, it is processed by an adaptive filter to smooth the blocks edges. A simplified block diagram of the decoder is shown in figure 1.

These video coding tools depend on the bitstream profile [6]. In the SBTVD [2], the transmission can be coded in baseline, main or high profile. In the following subsections, the most important features of the main profile are briefly described.

2.1. Reference picture lists management

Before the start of the decoding process of each slice, depending whether the slice is P or B, up to two reference lists are initialized addressing the pictures stored in the Decoded Picture Buffer (DPB). The DPB

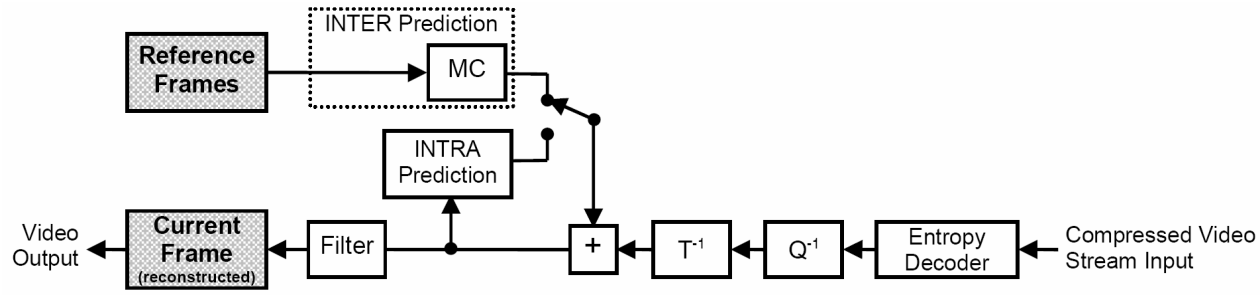

Fig. 1 - Simplified H.264/AVC decoder diagram [7]

has a limited space, depending on the level or on the maximum number of pictures stored, that is 16 pictures (16 frames or 32 fields), e.g.: for the level 4 (highest level in high profile at SBTVD) the pictures are limited by the total space of the buffer, so there are 4 reference pictures in the DPB and for the baseline, the pictures are limited by the maximum number of pictures (16), even if the space allows 24 reference pictures.

For the B-slices there are two lists, initially ordered by the Picture Order Count (POC), in the way that one list gives preference to the past pictures and the other to the future pictures in the display order. Since P-slices just have past reference pictures there is just one reference picture list initially ordered by the Picture Number (PicNum), one especial case of POC.

2.2. Intra prediction

The intra prediction mode is one of the novelties introduced into video coding by the H.264/AVC standard. Its algorithm explores the spatial redundancy of the image, where a block can be predicted from previously decoded blocks. Each predicted sample can be generated using a directional copy of border samples of the neighbor blocks. Such a directional copy can be as simple as a replication, the mean value of the border samples, or an N-tap filter of border samples.

For the luminance samples of a macroblock, the prediction can be performed over the entire macroblock, i.e. 16x16 samples, or in blocks of 4x4 samples. In the case of 4x4 block intra prediction, there are up to 9 possible prediction modes. For the 16x16 block intra prediction, up to 4 prediction modes are available. For the chrominance samples, the intra prediction modes are equivalent to those used for blocks of 16x16 luminance samples. The number prediction modes can be reduced according to the availability of neighbor samples.

2.3. Inter prediction

For the temporal (inter) prediction, the macroblocks can be divided into smaller partitions which can be 2 16x8, 2 8x16 or 4 8x8 luminance samples. The 8x8 partitions, on its turn, can be subdivided into subpartitions of 4x8, 8x4 or 4x4 samples. For each partition there might be up to two reference pictures for motion compensation. The motion compensation however is done at subpartition level, which can be down to 4x4 pixels, using a motion vector given for each reference picture of the current partition. Each partition has a corresponding one at the same position in the first list 1 reference picture, which is called the co-located partition.

The prediction of the motion vectors is done depending on the macroblock type, but the predicted motion vector is basically the median of the neighbors' partitions motion's vector already derived. The neighbors' partitions are the left one, the top one and the upper-right one, as shown in figure 2. If C partition is unavailable, D is used. There are, however, some particularities which makes the process more detailed such as motion vectors of partitions with different reference picture for motion compensation. The prediction is done for each reference picture used for the motion compensation of the partition, which can be up to two in a B-type macroblock.

In B type macroblocks, there is a direct prediction mode, for which no motion vector difference is transmitted. There are two types of direct prediction: spatial direct prediction and temporal direct prediction. For both direct prediction types, information about the co-located partition in the first list 1 reference picture is requested.

In the spatial direct prediction, if the co-located partition has a motion vector that is less than 1/2 luminance samples in magnitude, one or both list 0 picture and list 1 picture predicted motion vectors are set to zero. Otherwise, the predicted motion vector is derived as in normal prediction from the neighborhood.

In the temporal direct prediction, the list 0 picture and list 1 picture motion vector are derived as a scaled copy of the co-located partition motion vector. The list 0 picture motion vector of the predicted partition is the same as the co-located partition list 0 picture previously used for prediction, and the list 1 picture motion vector of the predicted partition is the first list 1 reference picture. The weighting factors for scaling each predicted motion vector are determined from the temporal distance of the pictures used for prediction and the current picture.

2.4. Context-based Adaptive Arithmetic Coding

The Context-based Adaptive Arithmetic Coding (CABAC) method is based on probability models for each

D	B	C
A	Current Partition	

Fig. 2 - Current partition neighborhood

syntactic element. A probability model is updated while coding syntactic elements, keeping local statistics. CABAC uses arithmetic coding with only two possible intervals, referred as MPS (most probable symbol) and LPS. Probability states store the value of the most probable symbol and its probability. Since the arithmetic engine deals only with two possible intervals, each data symbol must be binarized to encode only 0 and 1 decisions (bins).

The context model used for decoding each bin is derived based on previously decoded bins and syntactic elements.

2.5. Interlaced video coding

The H.264/AVC standard also supports interlaced video coding. This can be set either by coding entire slices as top or bottom fields, or at macroblock level in an adaptive frame. In an adaptive frame, the slice is processed in units of 16x32 luminance samples considered as macroblock pairs. The macroblock pairs can be coded either as two frames or two fields.

Interlaced coding features affects every block of the decoder, with several implications in the direct motion vector prediction and reference lists management. At the moment, the support for interlaced video is being studied by the group and has not been yet fully implemented in the software.

3. SOFTWARE IMPLEMENTATION

The main goal of the software was not to achieve the highest performance running on a PC, but to serve as a model for study, tests and validation of the hardware. Therefore, it was implemented with a separate source code file in an one to one correspondence to each of the hardware decoder modules. [4]

3.1. Code structure and algorithms

For global decoder control, there are structures that store and organize decoding information, e.g.: parameter sets, decoded samples, prediction modes of each macroblock, etc. The functions responsible for setting the decoding flow are described in a rather high abstraction level, in a way that function calls would be equivalent to enabling and feeding data into hardware modules.

The reference picture list is composed by pointers to where the pictures are in the DPB, since it is more efficient to deal with vectors of pointers. There are mechanisms to reorder the list, so the access to the desired pictures can be optimized. There are processes to mark the pictures (unused or used for reference) in a custom way.

For the inter prediction, memory should be allocated to store two motion vectors per 4x4 block of the image, regardless of macroblocks partitioning. Functions were created to predict the motion vectors of the subpartitions and predict the 4x4 blocks.

For the CABAC entropy decoder, there is a function to decode each syntactic element. The function calls gets the context model, perform the syntactic element anti-binarization and calls the arithmetic engine functions to decode the bins until a valid syntactic element value is obtained.

3.2. Validation tools

PRH.264 has a graphic user interface that allows the user to enable the debug tool for each module. While executing, the program checks for each option enabled and stores in a separate file intermediary data from the decoding process. These tools can be used to locate exact eventual mismatch points while comparing its outputs with data generated by the hardware decoder. Some tools were also used to validate the software itself.

For the inter prediction the program prints predicted motion vectors and motion vector difference obtained by the entropy decoder, reference pictures indexes, and each macroblocks predicted luminance and chrominance samples.

For intra prediction, PRH.264 prints into a text file the syntactic elements that determine the prediction mode, the macroblock address and the index of the 4x4 block inside the macroblock, when this type of prediction is used. The predicted samples of luma and chroma are also written in a text file for further analysis.

A modified version of the JM reference software 15.0 [5] was used for debug and validation of the CABAC entropy decoder written in C. The trace functionality in JM, was increased to give information about the decoding of each bin in the decoding process flow in the normal trace file and in a new file. This improved trace functionality was also implemented in PRH.264. Thus, the values of binary arithmetic engine interval range, interval offset and probability state could be compared. Using a file comparison tool, to get the first difference between PRH.264 trace file and the CABAC JM trace file, and searching where it occurs in the normal trace file, informed the frame number, macroblock address and syntactic element that was being decoded. This feature can be used to validate the hardware in a functional simulation.

4. RESULTS AND ANALYSIS

In order to extract relevant data from the model, the software was built to profile the code. A set of different H.264 bitstreams were encoded with JM reference software, configuring it to use the features present in the main profile.

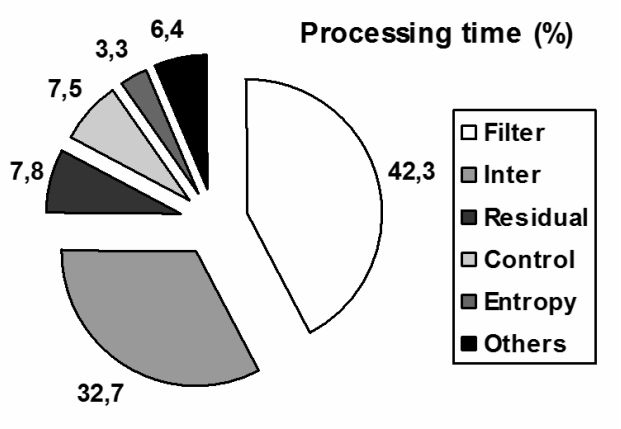

Fig. 3 - Dynamic analysis pie-chart

4.1. Dynamic analysis with video bitstreams

The video bitstreams were all generated with B slice inter prediction and CABAC. One video bitstream extracted from broadcasting which implemented only baseline features was also decoded. The results of frame-rate achieved by decoding them with PRH.264 in a PC Core 2 Duo E8400 3.0 GHz 4GB RAM DDR 800 are shown in the table 1.

Bitstream	Resolution	Frame rate	Length
Akyio	176x144	400 fps	150 frames
Bus	352x288	70 fps	150 frames
Parkrun	1280x720	7 fps	90 frames
Broadcast baseline	320x192	183 fps	>1000 frames

Tab. 1 - Frame rate results

The *gprof* tool was used to generate statistical data about the processing time spent in each function of the software while decoding the video bitstreams, achieving similar outputs. The results for the "bus" sequence are shown in the pie-chart of figure 3. From these functions, the time necessary to perform the most demanding decoding processes can be estimated and the module to which it applies can be identified.

The pie-chart shows that the deblocking filter demands higher processing time (approximately 42%), and functions related with motion compensation used to predict a single 4x4 block of the image (independently of the partitioning) and other used to get a 9x9 luminance sample for pixel interpolation, comes second with (approximately 33 %). Other functions like intra prediction, headers decoding with lesser execution times are related with 6.4 %

As a case of analysis, with a possible optimization of reducing the total execution time of functions related to the deblocking filter and motion compensation by half the total time needed to decode the video bitstreams would be reduced to approximately 70 %.

4.2. Identification of hardware design challenges

The most time-consuming processes in the software model were identified to be where a large amount of calculations with previously stored data must occur. Therefore, in these blocks, the hardware design should be optimized aiming at executing several operations in parallel. Also, the memory access should be carefully designed, considering the feasibility of implementing cache.

5. CONCLUSIONS AND FUTURE WORKS

Software modeling for systems design is a good approach when dealing with high complexity applications as H.264/AVC video decoding. Despite not giving an exact correspondence of the hardware behavior due to the fundamental difference in the methods of reaching high performance, a software model was successful to help the group to learn about the general decoder functioning. Also, the software was successfully used to generate test vectors to validate the modules of an intra-only FullHD hardware video decoder prototype.

As future work, the implementation that is being done in order do deal with interlaced video should be completed and tested. Then a few adaptations should be made for the software to decode video bitstreams at high profile. After the completion of these tasks, the software has to be validated with broadcasting videos in high definition. The final version of PRH.264 will be used to generate test vectors and validate the hardware decoder.

6. REFERENCES

[1] “Rede H.264 SBTVD Wiki”, www.lapsi.eletro.ufrgs.br/h264/wiki/, 2010.

[2] “Televisão Digital Terrestre – Codificação de video, audio e multiplexação”, ABNT, Rio de Janeiro-RJ, 2007.

[3] M.A Lorencetti, W.T. Staehler and A.A. Susin, “Reference C Software H.264/AVC Decoder for Hardware Debug and Validation”, XXIII South Symposium on Microelectronics, SBC, Bento Gonçalves-RS, pp 127-130, 2008.

[4] M.A. Lorencetti, W.T. Staehler, A.A. Susin, “Incremental Hardware Development from Modular Mixed C-VHDL Simulation”, 8th Students Forum on Microelectronics SForum’08, Gramado-RS, 2008.

[5] “H.264 Reference Software”, http://iphom.hhi.de/suehring/tml/, 2010.

[6] Video Coding Experts Group, “ITU-T Recommendation H.264 (03/05): Advanced video coding for generic audiovisual services”, *International Telecommunication Union*, 2005.

[7] I.E.G. Richardson, “H.264 and MPEG-4 Video Compression: Video Coding for the Next-generation Multimedia”, *John Wiley and Sons*, England, 2003.