2

Complexity/Performance Analysis of a H.264/AVC Video Encoder

Hajer Krichene Zrida[1], Ahmed Chiheb Ammari[2], Mohamed Abid[1] and Abderrazek Jemai[3]
[1]*Sfax University, ENIS Institute, Computer and Embedded Systems CES Laboratory,*
[2]*Carthage University, INSAT Institute, Research Unit in Materials Measurements and Applications (MMA),*
[3]*University of Tunis el Manar, Faculty of Science of Tunis, LIP2 Laboratory, Tunisia*

1. Introduction

The evolution of digital video industry is being driven by continuous improvements in processing performance, availability of higher-capacity storage and transmission mechanisms. Getting digital video from its source (a camera or a stored clip) to its destination (a display) involves a chain of components. Key to this chain are the processes of compression and decompression, in which bandwidth-intensive raw digital video is reduced to a manageable size for transmission or storage, then reconstructed for display (Richardson, 2003). The early successes in the digital video industry were underpinned by international standard ISO/IEC 13818 (ISO/IEC, 1995), popularly known as MPEG-2. Anticipation of a need for better compression tools has led to the development of the new generation H.264/AVC video standard. The H.264/AVC is aiming to do what previous standards did in a more efficient, robust and practical way, supporting widespread types of conversational (bidirectional and real-time video telephony, videoconferencing) and no conversational (broadcast, storage and streaming) applications for a wide range of bitrates over wireless and wired transmission networks (Joch et al., 2002).
The H.264/AVC has been designed with the goal of enabling significantly improved compression performance relative to all existing video coding standards (Joch et al., 2002). Such a standard uses advanced compression techniques that in turn, require high computational power (Alvarez et al., 2005). For a H.264 encoder using all the new coding features, more than 50% average bit saving with 1–2 dB PSNR (Peak Signal-to-Noise Ratio) video quality gain are achieved compared to previous video encoding standards (Saponara et al., 2004). However, this comes with a complexity increase of a factor 2 for the decoder and larger than one order of magnitude for the encoder (Saponara et al., 2004).
Implementing a H.264/AVC video encoder represents a big challenge for resource-constrained multimedia systems such as wireless devices or high-volume consumer electronics since this requires very high computational power to achieve real-time encoding. While the basic framework is similar to the motion compensated hybrid scheme of previous video coding standards, additional tools improve the compression efficiency at the expense

of an increased implementation cost. For this, the exploration of the compression efficiency versus implementation cost is needed to provide early feedbacks on the standard bottlenecks and select the optimal use of its coding features.
The objective of this chapter is to perform a high-level performance analysis of a H.264/AVC video encoder to evaluate its compression efficiency versus its implementation complexity and to highlight important properties of the H.264/AVC framework allowing for complexity reduction at the high system level. The complexity analysis focus mainly on computational processing time measures with instruction-level (Kuhn et al., 1998) profiling on a general purpose CISC Pentium processor. Processing time metrics are completed by memory cost measures as this have a dominant impact on the cost-effective realization of multimedia systems for both hardware and software based platforms (Catthoor et al., 2002), (Chimienti et al., 2002).
Actually, when combining the new coding features, the implementation complexity accumulates, while the global compression efficiency becomes saturated (Saponara et al., 2004). To find an optimal balance between the coding efficiency and the implementation cost, a proper use of the AVC tools is needed to maintain the same coding performance as the most complex coding parameters configuration (all tools on) while considerably reducing complexity. In this chapter, we will cover major H.264 encoding tools. Each new tool is typically tested independently comparing the performance and complexity of a complex configuration to the same configuration minus the tool under evaluation. The coding performance is reported in terms of PSNR and bit rate, while the complexity is estimated as the total computational execution time of the application and the maximum memory usage allocated by the source code. Absolute complexity values of the obtained cost-efficient configuration of the H.264 encoder shall confirm the big challenge of its cost-effective implementation using of a well-defined multiprocessor approach to share the encoding time between several embedded processors.
The chapter is organized as follows. The next section provides an overview of the new H.264 technical features. Section 3 defines the adopted experimental environment. The coding performance and complexity of the H.264 major encoding tools are evaluated in section 4. Section 5 shall give the complexity analysis, memory and task level profiling of an obtained cost-efficient configuration. Section 6 discusses some aspects related to previous parallelization studies for an efficient parallel implementation of this standard on a given multiprocessor platform.

2. Overview of the H.264/AVC video encoder

An important concept in the design of H.264/AVC is the separation of the standard into two distinct layers: a video coding layer (VCL), which is responsible for generating an efficient representation of the video data; and a network adaptation layer (NAL) (Richardson, 2003) which is responsible for packaging the coded data in an appropriate manner based on the characteristics of the network upon which the data will be used. This chapter is concerned with the VCL layer.

2.1 The coding layer block diagram

The block diagram of the video coding layer of a H.264/AVC encoder is presented in figure1. This figure includes a forward path (left to right) and a reconstruction path (right to left) (Richardson, 2003).

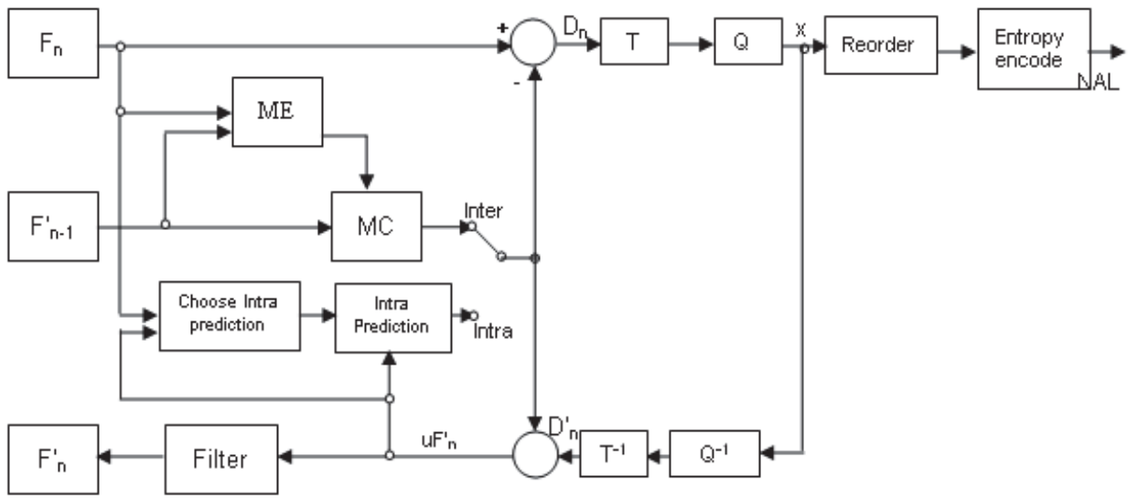


Fig. 1. H.264 /AVC video encoder block diagram

An input frame or field Fn is processed in units of a macro-block (MB). Each MB is encoded in intra or inter mode and, for each block in the MB, a prediction PRED (marked 'P' in figure1) is formed based on reconstructed picture samples. In Intra mode, PRED is formed from spatially neighboring samples in the current slice that have previously been encoded, decoded and reconstructed (uF'n in the figure1 note that unfiltered samples are used to form PRED). The encoding process chooses which and how neighboring samples are used for Intra prediction, which is simultaneously conducted at the encoder and decoder using the transmitted Intra prediction side information (Malvar et al., 2003).

In Inter mode, PRED is formed by motion-compensated prediction from one or multiple reference picture(s) selected from the set of reference pictures. In the figure1, the reference picture is shown as the previous encoded picture F'n-1 but the prediction reference for each MB partition (in inter mode) may be chosen from a selection of past or future pictures (in display order) that have already been encoded, reconstructed and filtered. The prediction PRED is subtracted from the current block to produce a residual difference block Dn that is transformed (using a block transform) and quantized to give X, a set of quantized transform coefficients which are reordered and entropy encoded. The entropy-encoded coefficients, together with side information required to decode each block within the MB (prediction modes, quantization parameter, motion vector information, etc.) form the compressed bit stream which is passed to a Network Abstraction Layer (NAL) for transmission or storage.

As well as encoding and transmitting each block in a MB, the encoder decodes (reconstructs) it to provide a reference for further predictions. The coefficients X are scaled (Q-1) and inverse transformed (T-1) to produce a difference block D'n. The prediction block PRED is added to D'n to create a reconstructed block uF'n a decoded version of the original block (u indicates that it is unfiltered). A filter is applied to reduce the effects of blocking distortion and the reconstructed reference picture is created from a series of blocks F'n.

2.2 Main innovations in comparison to previous standards

The basic functional elements of H.264 /AVC presented in figure1 represent a similar set of the generic DPCM/DCT (Richardson, 2003) coding and decoding functions of earlier standards. The H.264 provides higher coding efficiency through added features and

functionality that in turn entails additional complexity. Here we present a summary of the most relevant key features for the performance of this standard.

First, the motion compensation model supports the use of multiple reference frames for prediction with a weighted combination of the prediction signals. Also, it introduces variable block-size motion compensation with small block sizes that range from 16x16 up to 7 modes including 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 pixel blocks. Motion vectors can be specified with higher spatial accuracy with quarter-pixel and eighth-pixel instead of half-pixel accuracy. In order to estimate and compensate fractional-pel displacements, the image signal of the reference image has to be generated on sub-pel positions by interpolation. Pixel interpolation is based on a finite impulse response (FIR) filtering operation: 6 taps for the quarter resolution and 8 taps for the eighth one (Schäfer et al., 2003). A rate-distortion (RD) Lagrangian technique optimizes both motion estimation and coding mode decisions. Moreover, an adaptive deblocking filter is added to reduce visual artifacts produced by the block-based structure of the coding process (Ostermann et al., 2004).

For the intra-frame prediction, in contrast to previous video coding standards where prediction is conducted in the transform domain, prediction in H.264/AVC is always conducted in the spatial domain by referring to neighboring samples of already coded blocks (Schäfer et al., 2003). Two classes of intra coding modes are supported. When using the INTRA-4x4 class, each 4x4 block of the luma component utilizes one of nine prediction modes. Beside DC prediction, the standard supports eight directional prediction modes involving linear combinations of the samples. For the INTRA 16x16 classes, four prediction modes are supported (ISO/IEC, 2003).

The concept of Bipredictive (B) slices is generalized in H.264/AVC. B slices use a similar macroblock partitioning as for the Predicted (P) slices. This includes the Intra 4x4, the intra 16x16 all the inter 16 x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 modes. B slices are coded in a manner in which some macroblocks may use a weighted average of two distinct motion compensated prediction values, for building the prediction signal. Generally, B slices utilize two distinct reference picture buffers referred as the first and the second reference picture buffer, respectively. Four different types of inter prediction are supported: list0, list1, bi-predictive, and direct prediction. List 0 or List 1 prediction indicates that the prediction signal is formed by motion compensation from a picture of the first respectively the second reference buffer. In the bi-predictive mode, the prediction signal is formed by a weighted average of a motion-compensated list 0 and list 1 prediction signal. The direct prediction mode is inferred from previously transmitted syntax elements and can be either list 0 or list1 prediction or bi-predictive (Schäfer et al., 2003).

For the (T) transform, H.264/AVC employs a purely integer spatial approximation discrete cosine transform (DCT). This transform basically works on 4x4 shapes, as opposed to the conventional floating-point 8x8 DCT specified with rounding error tolerances that is used in earlier standards. The small size helps to reduce blocking and ringing artifacts, while the precise integer specification eliminates any mismatch between the encoder and decoder in the inverse transform (Ostermann et al., 2004). For the quantization (Q) of transform coefficients, H.264/AVC uses scalar quantization. The quantization step size is chosen by a so called quantization parameter QP which supports 52 different quantization parameters. One of 52 quantizers is selected for each macroblock by the Quantization Parameter (QP). The quantizers are arranged so that there is an increase of approximately 12.5% in the quantization step size when incrementing the QP by one (Malvar et al., 2003).

Finally, H.264/AVC specifies two alternative methods of entropy coding: a low-complexity technique based on the usage of context-adaptively switched sets of variable length codes, so-called CAVLC, and the computationally more demanding algorithm of context-based adaptive binary arithmetic coding (CABAC). Both methods represent major improvements in terms of coding efficiency compared to the techniques of statistical coding traditionally used in prior video coding standards (Ostermann et al., 2004).

3. Experimental environment

The complexity of the H.264 video encoder application depends on the algorithm, the encoding option tools, the input sequences and the architecture in which it is implemented. For making a complete analysis of the effect of the encoding option parameters on performance and complexity of a H.264 video encoding application, the JM encoder software reference version 10.2 is used with main profile @ level 4 (JM 10.2, 2005). Measurements have been done on a General-Purpose Processor (GPP) platform based on an INTEL Centrino 1.6 GHZ running a Linux operating system.

The encoding option parameters are representative of the standard encoding new tools. For this analysis, each coding tool is tested independently comparing the performance and complexity of a complex configuration to the same configuration minus the tool under evaluation. For the starting complex configuration, a full search algorithm for motion estimation is fixed, P (predicted) and B (Bi predicted) frame weighted prediction is used, motion vectors fractional pixel accuracy is applied with variable block sizes supported (7 motion compensation block types) and multi-frame references fixed to 5. A loop filter and Hadamard transform are used. The Rate-Distortion (R-D) optimization technique with an explicit Lagrangian parameter selection is activated. The input search range is fixed to 32, and the quantization parameter (QP) values is fixed to 28 for I and P slices, 30 for B slices and 29 for B reference slices. For B frame generalization, only one reference is used for list0 and list1. Motion estimation based on the spatial direct and bi-predictive modes is thus activated. The CABAC entropy method is used.

For video streaming and video conferencing applications, we used popular test video sequences in the Common Intermediate Format (CIF, 352 × 288 picture elements) and in the Quarter Common Intermediate Format (QCIF, 176×144 picture elements). 7 test sequences in a 4:2:0 YUV format with different grades of motion characteristics and frame rate (trace.eas, n.d.) are used as given in table1. “Bridge far”, “container” and “Mother & Daughter” offer a wide variety of video QCIF content occurring in low-bit-rate applications of tens of Kbps. “Foreman” is a good medium complexity QCIF test sequence for medium bit rate applications of hundreds of Kbps. The CIF version of “Paris” and “Bridge close” are useful test cases for middle-rate applications. Finally, “Mobile” is a high-complexity CIF sequence with lot of movements including rotation and is a good test for high-rate applications of thousands of Kbps.

4. H.264/AVC performance and complexity parametric analysis

In this section, the coding performance and complexity of the H.264 major encoding tools are evaluated. The coding performance is reported in terms of PSNR and bit rate output, while the complexity metrics focus mainly on the amount of computing time required to encode a given test sequence on the used GPP platform. As motion estimation is the most

important computing part of the encoder, the computing complexity of this module is particularly noted for all the experimented simulations. Processing time metrics are completed by memory cost measures as this have a dominant impact on the cost-effective realization for both hardware and software based platforms.

Sequence	Format (Pixel)	Frame Rate (Hz)	Frames Coded
Bridge close	CIF (352x288)	15	2000
Mobile	CIF	25	300
Paris	CIF	15	1065
Bridge far	QCIF (176x144)	15	2101
Container	QCIF	25	300
Foreman	QCIF	25	400
Mother & Daughter	QCIF	25	961

Table 1. Used test video sequences

4.1 Coding structures influence evaluation

The influence of the different H.264 encoding structures, including the classical coding types and the advanced pyramid coding structures is analyzed. In this section, only the first 150 frames of all the test sequences are used. This shall provide the best optimal coding order for the best encoding performance. The used structures are described as follows:

- An I-P-P-P-P… coding and display order using P only coding,
- an I-B-P-B-P… coding order with one non reference B slice,
- an I-B-B-P-B-B-P… coding order with 2 non reference B slices,
- an I0-P4-RB2-B1-B3-P8… coding order with 3 level pyramid using 3 B pictures (3L3B),
- an I0-P6-RB2-RB4-B1-B3-B5-P12… coding order with 3 level pyramid using 5 B pictures (3L5B),
- an I0-P8-RB2-RB4-RB6-B1-B3-B5-B7-P16.. coding order with 3 level pyramid using 7 B pictures (3L7B),
- an I0-P8-RB4-RB2-RB6-B1-B3-B5-B7-P16.. coding order with 4 level pyramid using 7 B pictures (4L7B),
- and an I0-P12-RB6-RB3-B1-B2-B4-B5-RB9-B7-B8-B10-B11-P24… coding order with 4 level pyramid using 11 B pictures (4L11B).

Bit rate output performance results are presented in figure 2 for four selected sequences. This figure indicates clearly that the bit rate output is significantly improved using reference B slices (up to 35% bit rate reduction with one non reference B slice and 15% more bit rate reduction with two non reference B slices for the CIF version of “Bridge-close”). The bit rate output and the PSNR video quality are better using Pyramid structures compared to the classical coding structures (between 5 and 10% bit rate reduction with a light PSNR improvement with 3L3B, and much better with 3L5B and 3L7B). For this, making the use of these pyramid structures is interesting. According to the obtained results, the best structure in term of coding performance is 3Level-7B pyramid. However, compared to 3Level-5B pyramid structure, the 3Level-7B requires more computational time for practical the same performance. Thus, to achieve the best performance with a minimum complexity, the

3Level-5B pyramid is preferred. The 4Level-7B pyramid and the 4Level-11B pyramid don't appear to provide any additional performance compared to the 3Level-5B pyramid as a small performance loss in bit rate is observed.

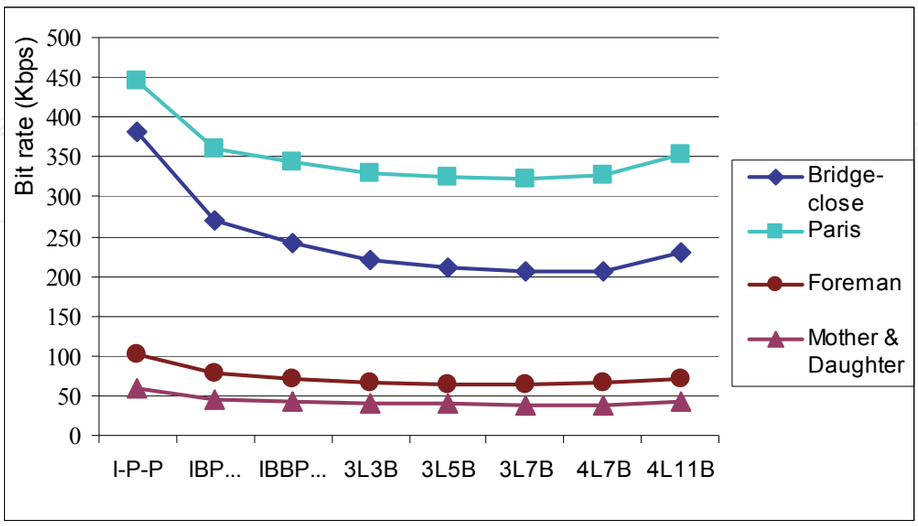


Fig. 2. Bit rate for various coding structures and video format

Given these obtained results, it is clear that the 3Level-5B hierarchical coding order offers the best performance/complexity values. Given this, the 3Level-5B is the adopted structure for the starting complex configuration.

4.2 Performance and computing complexity of the reference configuration

Performance and computing complexity of the H.264 complex reference encoder configuration is first estimated for all the test sequences of table 1. Results of this analysis are reported in table 2 as the total processing time, the motion estimation ME time, the bit rate output, and the luminance PSNR values. The PSNR values, given in dB, are representative of the obtained encoding performance. More the PSNR value is high, more the image quality and the encoding performance are better. Given these results, it is obtained that even for the low-bit-rate QCIF "bridge far" sequence, the time required to compute the encoding algorithms on the GPP platform is of 5137.08 second. The associated encoding performance in frames per second is of 0.41 fps. Really, this is too far from a real time video encoding performance of 25 frames per second. As a consequence, an optimal selection of the new coding tools can allow for roughly the same performance as for the complex reference configuration but with a considerable complexity reduction.

4.3 Performance and computing complexity of major encoding tools

This section presents a performance and computing complexity analysis of some major encoding tools. The considered tools are the search size, the variable block size, the multiple reference frames, the fractional pixel accuracy, and the bi-prediction motion estimation. The efficiency of the fast motion estimation algorithms, the R-D Lagrange technique, the Hadamard transform and the entropy coding techniques are also evaluated. To find an optimal trade-off between coding efficiency and implementation complexity, the effect of

each coding tool is tested separately in comparison with the fixed reference configuration. We will observe varying complexity values at a gain in the obtained video quality and bit-rate output.

Res.	Sequence	Total time (s)	ME Time (s)	ME Complexity (ME C %)	frames per Seconds (fps)	Bit rate (Kbps)	PSNR-Y (dB)
CIF	Bridge-close	19259,41	15670,33	81,36	0,1	106,44	35,01
	Mobile	3027,4	2343,04	77,39	0,1	676,08	32,82
	Paris	9479,15	7327,81	77,3	0,11	129,54	35,28
QCIF	Bridge-far	5137,08	4295,38	83,62	0,41	2,74	37,84
	Container	715,06	580,44	81,17	0,41	19,37	36,17
	Foreman	1026,86	838,6	81,67	0,39	79,2	35,01
	Mother & Daughter	2145,51	1728,42	80,56	0,45	30,32	36,3

Table 2. Performance/complexity of the reference configuration

4.3.1 Full/Fast full motion estimation

The full Search motion estimation is reported to be the most-consuming part of the entire encoding process (Pascalis et al., 2004). For this, several fast motion estimation algorithms have been proposed (Pascalis et al., 2004), (Chen et al., 2002). In our case, the efficiency of the UMHexagonS fast search algorithm (Chen et al., 2002) is analyzed in comparison with the full search estimation scheme. The obtained results of this analysis are reported in table 3. It is clear from table 3 that using the UMHexagonS search method we got a very slight bit rate and PSNR degradations in comparison with a full search algorithm. But, this comes with up to 45% of computation time complexity reduction. Thus, as the fast full search technique considerably improves the coding complexity without a notable loss in video quality and bit rate for all test sequences, the UMHexagonS will be adopted as a fast motion estimation scheme.

4.3.2 Search range

The influence of the search range (SR) window is evaluated for different SR values. The obtained results are given in table 4 as the total processing time, the bit rate output, and PSNR values. As shown in table 4, an important complexity reduction is obtained using a search range of 8 compared to 16 and 32 values, at a cost of a negligible loss in bit rate and video quality. For consequence and for a cost-efficient configuration, a search range of 8 is chosen.

4.3.3 Variable block sizes

The influence of three block size modes is evaluated. The first mode is with 7 block sizes activated (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4), the second is with 4 (16x16, 16x8, 8x16, and 8x8), and the third is with only one 16x16 block size. As presented in table 5, supporting all the seven block sizes increases the computational complexity especially for the motion estimation module, at a gain in the coding efficiency. Compared, with the 4 block size

(16x16, 16x8, 8x16, and 8x8), we got a light video quality degradation with a negligible loss in bit rate (negligible loss for the QCIF version of "bridge far" and less than 2.5% for the CIF version of "mobile"), but with a 10% average complexity reduction. With only one 16x16 block size mode, we got more significant video quality degradation compared to that with four block sizes, but with an encoding time further reduced 10% average. These results confirm that block sizes smaller than 8x8 (i.e. the seven block size mode on) do not provide significant benefits compared with the 4 block size mode. However, with the use of only 16x16 block size, the encoding performance is significantly decreased. For consequence, to reduce the implementation complexity while maintaining the same encoding performance, the 4 block size mode is adopted.

Resolution		CIF			QCIF			
ME Algo	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
Full Search	ME C (%)	81,36	77,39	77,3	83,62	81,17	81,67	80,56
	Bit rate	106,44	676,08	129,54	2,74	19,37	79,2	30,32
	PSNR-Y	35,01	32,82	35,28	37,84	36,17	35,01	36,3
Fast ME	ΔTEC (%)	-39,43	-32,33	-40,16	-41,66	-40,03	-38,53	-42,92
	ME C (%)	70,26	66,28	62,81	72,81	69,19	71,41	66,43
	ΔBit rate	-0,03	1,43	0,8	-0,01	0,08	0,08	-0,04
	ΔPSNR-Y	-0,02	0	-0,02	-0,03	0	-0,02	-0,03

ΔTotal Encoding Complexity (ΔTEC (%)) = Encoding Complexity (with Fast ME algorithm) – Encoding Complexity (with Full Search). ΔBit rate (Kbps) = Bit rate (with Fast ME algorithm) – bit rate (with Full Search), idem for ΔPSNR-Y

Table 3. Performance and Complexity Results for Full Search and Fast Full Search Algorithms

Resolution		CIF			QCIF			
Search Range	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
32	ME C (%)	70,26	66,28	62,81	72,81	69,19	71,41	66,43
	Bit rate	106,41	677,51	130,34	2,73	19,45	79,28	30,28
	PSNR-Y	34,99	32,82	35,26	37,81	36,17	34,99	36,27
16	ΔTEC (%)	-42,68	-42,58	-37,46	-40,63	-43,09	-43,61	-39,68
	ME C (%)	47,52	41,91	40,22	53,21	47,36	49,57	44,9
	ΔBit rate	0	-0,12	0,16	-0,01	-0,03	0,35	0
	ΔPSNR-Y	0,01	-0,01	0	0	0	-0,01	0,01
8	ΔTEC (%)	-23,98	-19,87	-19,85	-28,73	-23,45	-24,80	-20,26
	ME C (%)	30,72	26,38	25,78	36,4	29,81	33,42	30,94
	ΔBit rate	0,26	1,07	0,73	0,01	0	1,81	0,08
	ΔPSNR-Y	-0,01	0	0	0	0	0	-0,01

Table 4. Performance and complexity results for various search sizes

Resolution		CIF			QCIF			
Block Sizes	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
7	ME C (%)	30,72	26,38	25,78	36,4	29,81	33,42	30,94
	Bit rate	106,67	678,46	131,23	2,73	19,42	81,44	30,36
	PSNR-Y	34,99	32,81	35,26	37,81	36,17	34,98	36,27
4	ΔTEC (%)	-11,32	-10,51	-10,23	-9,11	-9,16	-11,91	-11,26
	ME C (%)	27,42	22,21	22,34	33,81	26,71	29,31	26,99
	ΔBit rate	0,67	14,53	5,19	0	0,45	1,32	0,18
	ΔPSNR-Y	-0,06	-0,11	-0,13	-0,03	-0,09	-0,09	-0,14
1	ΔTEC (%)	-11,62	-14,17	-11,24	-13,14	-12,33	-13,87	-12,75
	ME C (%)	25,67	19,11	19,98	32,8	25,14	26,73	24,58
	ΔBit rate	2,3	58,35	15,36	0,01	2,96	9,15	2,82
	ΔPSNR-Y	-0,09	-0,14	-0,17	-0,07	-0,18	-0,2	-0,22

Block sizes=7, then all seven modes (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4) are on.
Block sizes=4, then 16x16, 16x8, 8x16, and 8x8 modes are on.
Block sizes=1 only 16x16 mode is on

Table 5. Performance and complexity results for motion compensation blocks sizes

4.3.4 Multiple reference frames

Results concerning the influence of the multiple reference frame option are reported in table 6. Using this table, we observe for example for the CIF "bridge close" an increase of 43% bit rate for a reference frame number reduction from 5 to 1. This goes also for the QCIF "Foreman" video sequence with a 50% of bit rate increase for also a reference frame reduction from 5 to 1. However, with the use of only 3 reference frames, we observe a slight gain in the computational complexity and less than 5% bit rate increase with a little video quality degradation. Thus, using only 3 reference frames leads to a somewhat computational burden decrease without a noticeable coding efficiency degradation compared to that obtained with 5 reference frames. However, using only one reference frame leads to a sensible loss in coding performance with a slight complexity reduction. Thus, the optimal reference frame number is fixed to 3 for an optimized configuration.

4.3.5 RD-Lagrangian optimization

The R-D optimization is the criterion for selecting the best coding mode. It evaluates the cost of every possible coding mode, considering the balance of the distortion and the number of consumed bits. The obtained mode with the smallest cost will be considered as the best coding mode. As presented in table 7, the R-D Lagrangian technique gives a substantial compression efficiency improvement at a double complexity cost. The encoder without RD optimization is about 2~3 times faster and gives a noticeable loss in bit rate-distortion compared to the case with an RD-Lagrangian technique enabled (an average of 40% in bit rate increase in case of QCIF "bridge far" sequence, as described in table 7). While the considerable computational complexity required by the R-D optimization, it is a very

important tool of the JM reference software. As our objective is to obtain comparable performance as for the reference configuration, this option will be maintained.

4.3.6 Hadamard transform

A Hadamard transform may be used to improve the error cost functions performance such as the sum of absolute differences (SAD). However, given the obtained results of table 8, activating the Hadamard transform causes a slight complexity increase without any coding efficiency gain. Thus, the Hadamard transform will be disabled for the optimized parameter configuration.

Resolution		CIF			QCIF			
Reference Frames	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
5	ME C (%)	27,42	22,21	22,34	33,81	26,71	29,31	26,99
	Bit rate	107,34	692,99	136,42	2,73	19,87	82,76	30,54
	PSNR-Y	34,93	32,7	35,13	37,78	36,08	34,89	36,13
3	ΔTEC (%)	0,59	-1,78	0,05	0,30	-0,91	1,15	0,79
	ME C (%)	27,68	22,81	22,57	33,55	27,13	30,23	27,6
	ΔBit rate	6,7	26,98	6,47	0	0,81	2,59	1,15
	ΔPSNR-Y	-0,02	-0,03	-0,02	0	-0,03	-0,04	-0,04
1	ΔTEC (%)	0,19	1,80	0,68	-0,43	-0,21	3,89	2,53
	ME C (%)	27,81	24,12	22,91	33,9	27,61	30,79	27,98
	ΔBit rate	39,5	285,65	50,9	-0,08	5,27	41,43	12,44
	ΔPSNR-Y	-0,1	-0,43	-0,19	-0,01	-0,27	-0,43	-0,36

Table 6. Performance and complexity results for multiple reference frames

Resolution		CIF			QCIF			
RD-Lagrange	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
Enabled	ME C (%)	27,68	22,81	22,57	33,55	27,13	30,23	27,6
	Bit rate	114,04	719,97	142,89	2,73	20,68	85,35	31,69
	PSNR-Y	34,91	32,67	35,11	37,78	36,05	34,85	36,09
Disabled	ΔTEC (%)	-61,65	-68,63	-66,85	-53,81	-59,31	-59,61	-60,69
	ME C (%)	74,28	73,26	71,26	78,43	70,66	76,48	74,68
	ΔBit rate	23,34	152,46	13,93	1,09	2,51	14,3	5,1
	ΔPSNR-Y	0,24	0,32	0,06	-0,08	-0,11	0,04	0,01

Table 7. Performance and complexity results for R-D Lagrangian technique

Resolution		CIF			QCIF			
Hadamard	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
Enabled	ME C (%)	27,68	22,81	22,57	33,55	27,13	30,23	27,6
	Bit rate	114,04	719,97	142,89	2,73	20,68	85,35	31,69
	PSNR-Y	34,91	32,67	35,11	37,78	36,05	34,85	36,09
Disabled	ΔTEC (%)	-3,25	-2,10	-1,41	-3,92	-2,53	-2,60	-2,93
	ME C (%)	25,29	20,6	20,47	31,38	25,09	27,65	24,89
	ΔBit rate	0,12	1,49	0,76	0	0	0,44	0,05
	ΔPSNR-Y	-0,01	-0,04	-0,05	0,01	-0,03	-0,06	-0,07

Table 8. Performance and complexity results for Hadamard transform

Resolution		CIF			QCIF			
	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
With Fractional Pixel Accuracy	ME C (%)	25,29	20,6	20,47	31,38	25,09	27,65	24,89
	Bit rate	114,16	721,46	143,65	2,73	20,68	85,79	31,74
	PSNR-Y	34,9	32,63	35,06	37,79	36,02	34,79	36,02
Without Fractional Pixel Accuracy	ΔTEC (%)	-4,60	-3,98	-6,25	-7,05	-5,87	-6,08	-7,09
	ME C (%)	21,52	16,52	16,65	26,81	20,62	22,44	20,13
	ΔBit rate	2,84	452,18	25,32	0,02	9,06	26,96	9,62
	ΔPSNR-Y	-0,14	-0,45	-0,12	-0,05	0	-0,34	-0,24

Table 9. Performance and complexity results for fractional pixel motion compensation accuracy

4.3.7 Fractional pixel motion compensation

According to table 9, disabling the fractional pixel motion compensation accuracy option results in a significant increase of the bit rate output (more than 30% of bit rate increase for the QCIF "foreman" sequence and about 63% for CIF "mobile" sequence), with a light video quality degradation and a 5% average gain in complexity reduction. Thus, in order to maximize the coding performance, the fractional pixel accuracy option should be activated.

4.3.8 Bi-prediction motion estimation

Given results of table 10, disabling the bi-prediction motion estimation tool leads to a 20% average complexity reduction, without any noticeable coding efficiency degradation in terms of bit rate output and PSNR video quality. Thus, the use of bi-prediction motion estimation does not provide any significant improvement in the compression efficiency for

all the tested CIF and QCIF sequences. So, the bi-prediction motion estimation option shall be disabled.

4.3.9 Entropy coding

Given the results of table 11, it is clear that the CABAC entropy coding method provides noticeable gains in coding efficiency. Typically, it offers, for many sequences, between 5 to 10 percent efficiency gain and larger gains for higher resolution sequences. This comes with noticeable complexity drawbacks. However, The CAVLC entropy method offers much more implementation simplicity and offer about 25% of complexity reduction, with only a slight bit rate increase. Thus, for an optimized complexity configuration, CAVLC entropy coding method will be used.

Resolution		CIF			QCIF			
Bi-Predict ME	Seq.	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
Enabled	ME C (%)	25,29	20,6	20,47	31,38	25,09	27,65	24,89
	Bit rate	114,16	721,46	143,65	2,73	20,68	85,79	31,74
	PSNR-Y	34,9	32,63	35,06	37,79	36,02	34,79	36,02
Disabled	ΔTEC (%)	-21,29	-17,58	-19,27	-28,81	-23,13	-23,32	-24,31
	ME C (%)	6,45	6,59	5,58	7,2	6,29	8,76	7,32
	ΔBit rate	0,01	6,4	0,81	0	0,05	1,32	0,07
	ΔPSNR-Y	-0,01	-0,04	0	0	0	-0,03	0

Table 10. Performance and complexity results for bi-prediction motion estimation

Resolution		CIF			QCIF			
Entropy Coding Method	Sequence	Bridge-close	Mobile	Paris	Bridge-far	Container	Foreman	Mother & Daughter
CABAC	ME C (%)	6,68	6,74	5,75	7,39	6,33	8,89	7,33
	Bit rate	113,95	727,49	143,5	2,51	20,92	82,88	31,11
	PSNR-Y	34,89	32,59	35,04	37,79	35,94	34,76	36,06
CAVLC	ΔTEC (%)	-24,60	-24,61	-26,58	-21,54	-26,94	-23,76	-24,89
	ME C (%)	8,66	9,28	7,62	9,46	8,29	11,66	9,56
	ΔBit rate	8,31	37,63	6,43	0,07	1,11	5,11	2,08
	ΔPSNR-Y	0,02	-0,06	0,02	0,05	-0,04	-0,02	-0,01

Table 11. Performance and complexity results for the two entropy coding methods

4.4 Memory cost analysis

The data dominance of a video system implies that the memory cost have a dominant impact on the realization efficiency (Denolf et al., 2002). Application specific hardware implementations have to match memory system to the application. An efficient design flow uses this to reduce area and power. Thus, providing for the H.264/AVC a high level analysis of memory cost is essential to identify its resource requirements for hardware and software platforms. For each test sequence and for all the previously reported H.264 configurations, peak memory usage is measured using the "memprof" GNU profiler (memprof, n.d.). The obtained peak memory usage dependencies are reported in table 12. It is obtained that the encoder peak memory usage depends on the video format and linearly on the number of reference frames and the search size. The influence of the other coding tools and the input video characteristics is negligible.

Search size	QCIF			CIF		
	1F	3F	5F	1F	3F	5F
32	5.68	10.52	15.52	10.74	18.68	26.6
16	2.87	5.02	7.1	7.92	12.92	18.23
8	2.15	3.59	4.93	7.13	11.81	16.08

Table 12. Memory cost (in Mb) for different video formats, search size and reference frames

5. Complexity analysis of the optimized configuration

Given the previous analysis, the optimized configuration is presented as follows. A 3L5B pyramid coding structure, an UMHexagonS fast motion estimation scheme, a search range fixed to 8, 4 variable block sizes, 3 reference frames, R-D Lagrangian optimization activated, Hadamard transform disabled, motion vector fractional pixel accuracy enabled, P and B frames weighted prediction with bi-prediction motion estimation disabled, a QP value fixed to 28, and CAVLC entropy coding technique used.

5.1 Performance/computing time complexity

For this final configuration, the encoding performance and computing time complexity are obtained and given in table 13. In comparison with results of table 2, one order of magnitude in complexity reduction has been achieved with less than 10% average bit rate increase for all the CIF and QCIF used video test sequences. However, for this optimized configuration and even for the very low bit rate QCIF "bridge far" sequence, the time required to compute the encoding algorithms on the GPP platform is of 597.87 second. The associated complexity in frames per second is of 3.51 fps. Even with this configuration offering an optimal trade-off between coding efficiency and implementation complexity, we are still very far from a real time performance of 25 frames per second. Implementing this configuration of the encoder represents a big challenge for resource-constrained multimedia systems such as wireless devices or high-volume consumer electronics since this requires very high computational power to achieve real-time encoding.

Res.	Sequence	Total time (s)	ME Time (s)	ME C (%)	frames per Seconds (fps)	Bit rate (Kbps)	PSNR-Y (dB)
CIF	Bridge-close	2463,25	213,42	8,66	0,81	122,26	34,91
	Mobile	488,26	45,3	9,28	0,6	765,12	32,53
	Paris	1451,57	110,57	7,62	0,73	149,93	35,06
QCIF	Bridge-far	597,87	56,54	9,46	3,51	2,58	37,84
	Container	91,71	7,6	8,29	3,22	22,03	35,9
	Foreman	130,24	15,19	11,66	3,05	87,89	34,71
	Mother & Daughter	289,85	27,71	9,56	3,32	33,19	36,05

Table 13. Performance/computing time complexity of the reference configuration

5.2 Memory profiling

For the optimized configuration, the peak memory cost is of 5.02 MB for the QCIF and 12.92 MB for the CIF sequences. Comparisons with MPEG 4 Part2, simple profile with a 16 search size, half pixel resolution and I and P pictures are provided in (Saponara et al., 2004). For the memory usage, MPEG4 requires 2.97 MB for the QCIF and 9.88 for the CIF sequences. This result refers to no optimized MPEG4 source code. Applying platform independent memory optimizations through C level code transformations may be used to get a memory and algorithmic optimized version of the reference code. An example of such optimizations is applied in (Denolf et al., 2000) for an MPEG4 simple profile video decoder and in (Vleeschouwerand et al., 2001) for an encoder. By applying such optimization techniques, an optimized MPEG 4 simple profile is obtained using only 348.2 Kb of memory for CIF sequences (Vleeschouwerand et al., 2001). This represents a memory decrease with a factor of 30.

These memory optimizations can also be applied to our AVC optimized configuration. However, for the AVC case, the number of B frames is not limited to one B between two I/P frames, thus the memory compactation transformations used in (Vleeschouwerand et al., 2001) become invalid. Actually, even with possible optimizations, still around a minimum of few MB would be required, which is a problematic size for a realistic implementation.

Memory profiling of this optimized configuration is shown in figure 3. This figure presents the memory usage distribution over the main modules of the encoder. The "Init_Motion_Search_module" for the motion estimation is the most memory consuming with 67% of the total memory usage.

5.3 instruction-level profiling

For the 300 frames QCIF "Container" sequence and using the H.264/AVC encoder with the optimized configuration, we have performed an analysis of dynamic instruction distribution using the "Iprof" GNU profiler (Kuhn, 1999). The obtained results are shown in the following figure 4. It is clear from this figure that the H.264/AVC is dominated by integer operations, most of them are add, sub and shift instructions. Given the lot of data transfer operations, there are more memory instructions (more of 41%) than effective computation ones.

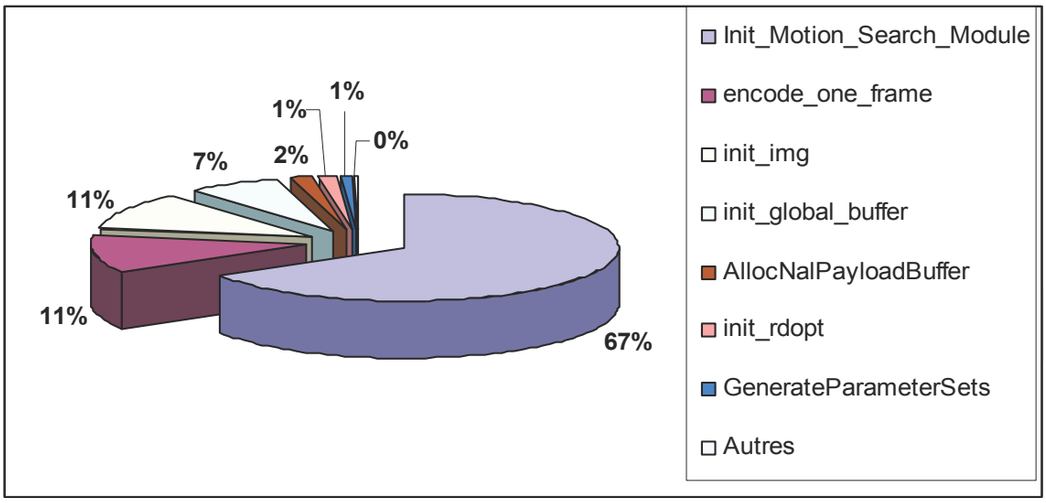


Fig. 3. Memory profiling of the optimized encoder configuration

```
                  iprof 0.4.2
                   ---------
   ... the portable Instruction Level Profiler
         (c) Peter Kuhn, 1996, 1997, 1998
Statistics File: bb.out
Executable File: lencod.exe
File Format:     elf32-i386
Architecture:    x86
Header of bb.out: Basic block profiling finished
 on Thu May 10 01:18:21 2007
Number of Basic blocks total: 14556

Instruction Usage Statistics: 0
----------------------------
  0 push              8571320898   (4.77338 %)
  1 mov              73746749854  (41.06965 %)
  2 sub               5059901750  (2.81787 %)
  3 call              2454848755  (1.36711 %)
  4 ret               1199273682  (0.66788 %)
  5 pushl                  256  (0.00000 %)
  6 jmp                836389570  (0.46579 %)
  7 add              19034677293  (10.60043 %)
  8 xor                649757875  (0.36185 %)
  9 pop               3207175757  (1.78608 %)
 10 and                827006618  (0.46056 %)
 11 nop                255745587  (0.14243 %)
 12 test              2924534178  (1.62868 %)
 13 je                3407442640  (1.89761 %)
 14 cmpl              6218794987  (3.46325 %)
 ..
Total Instructions used:    179565059320 (100%)
                              = 179565.05932000
Million
```

Fig. 4. Instruction breakdown of the optimized encoder configuration

Taken that the instruction per cycle is given by **IPC=InstCount/(Feq*ExecTime)**, for the used 1.6 GHz clock frequency GPP machine, and with an obtained number of instructions per frame of 598.55 106 (179565.059106 / 300), the obtained IPC is of 0.92. For a higher performance 3.0 GHz GPP machine, the necessary IPC for encoding H.264/AVC in real time should be 4.98. From these results we can note that even with a high frequency 3.0 GHz processor, approximately 5 instructions per cycle have to be executed to achieve H.264 real time encoding QCIF video sequences. Thus, using a single processor to real time encode H.264 bit streams may require a very high performance, high frequency super scalar processor. Such a choice is not suitable for embedded systems that have strict power and cost constraints.
An alternative solution is to use a multiprocessor approach to share the encoding execution time between several embedded processors. The sequential encoder application has to be distributed using a parallel programming model over a multiprocessor architecture. Based on that, we can conclude that it is necessary to explore multiple ways of parallelization apart from SIMD extensions in order to achieve the required performance for real time operation. To find the best scheme for parallel code execution, profiling the execution of the obtained configuration shall identify the major application bottlenecks and the main subcomponents candidate for efficient parallelization.

5.4 Execution profiling

Typically, tasks will not need the same amount of processing time. Thus, a computational profiling should be considered to identify the most computationally-expensive tasks and to give a clear picture of the critical code parts candidate for task-level parallelization. After that, complex tasks may also be subdivided further into smaller ones, i.e. each slowest compute node must be split in a set of compute nodes with better execution values.
For this, we have profiled the execution of the 300 frames of QCIF "Container" sequence with the "Gprof" GNU profiler (Graham et al., 1982). The obtained results are reported in the following figure 5 in terms of the CPU time percentage spent in the execution of each module. The obtained profile shows that the motion estimation and compensation (MEC), DCT transform, the entropy coding, the rate-distortion optimization (RDO) intra/inter mode decision, and the intra-prediction modules are the most time-consuming modules. These tasks constitute the major bottlenecks of the encoder.

6. Parallelization of the H.264/AVC video encoder

In the previous sections, we motivated the implementation of H.264/AVC encoder application on a multiprocessor platform. Actually, using a single processor to real time encode H.264/AVC bit streams may require a high performance, high frequency super scalar processor. Such a choice is not suitable for systems that have strict power and cost constraints. For such case, it may be probably necessary to use some kind of multiprocessor approach to share the encoding application execution time between several processors.
For the cost-efficient H.264/AVC parameters configuration, the obtained absolute complexity values and profiling analysis results confirmed the big challenge needed for a parallel multiprocessor execution. Parallelization consists in transforming the sequential encoding algorithms into concurrent tasks for execution in a multiprocessor system (Li et al., 2005). The predominant forms of parallelism in such systems are data-level parallelism (DLP) and task-level parallelism (TLP). DLP is perhaps the most commonly used form of parallelism, implemented through vector or SIMD architectures. The benefits of TLP are achieved by

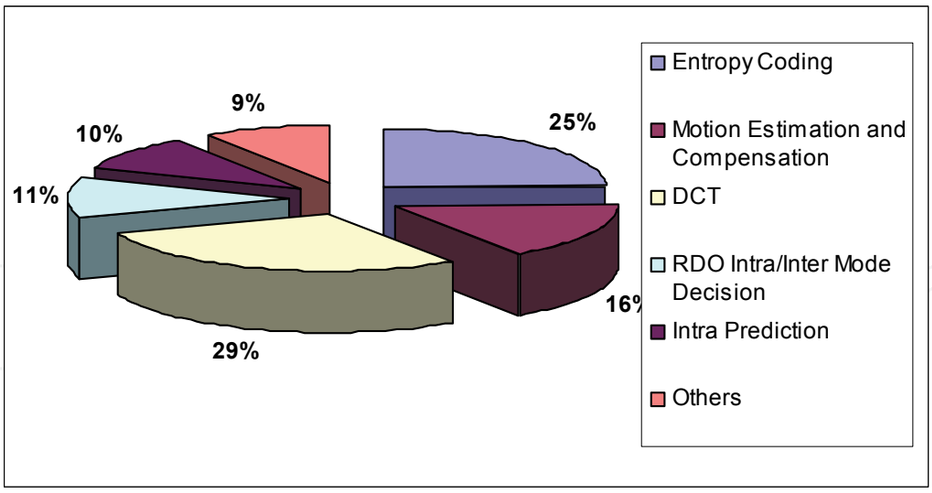


Fig. 5. Computational profile of H.264 video encoding

distributing the workload of a single high performance processor among a number of slower and simpler processor cores. This requires first to split the algorithms into separate tasks that may be executed at the same time, then to establish the necessary inter-task communication using parallel programming model primitives (Youssef et al., 2004). Generally, the parallel task execution is limited by data dependency between tasks. A data dependency means that one task needs the result of another one to be processed therefore limiting ways for parallelization (Pastrnak et al., 2006).

Given this, several multiprocessor and multi-threading encoding systems and parallel implementation methodologies have been proposed and discussed in many previous research studies (Gulati et al., 2005; Chen, 2004; Zhao, 2006; Sun, 2007) to find the best parallel execution scheme of the H.264/AVC video encoder for a chosen multiprocessor platform. Based on the performance results obtained in these previous works, and given our concern with resource constrained devices, we developed in a dedicated work a new high-level independent target-architecture parallelization approach (Krichene Zrida et al., 2009) based on the use of the parallel streaming programming models of computation and the simultaneous exploration of the two predominant concepts of parallelism; the data-level partitioning and the task-level splitting and merging. The goal of this approach is to derive in a structured way a parallel model of the encoder with the best computation and communication workload balance. Based on this parallelization approach (Krichene Zrida et al., 2009), a starting parallel model of the H.264/AVC reference encoder is first proposed. The implementation of this model is performed according to an appropriate programming strategy. According to the communication and computation concurrency properties of the implemented starting model, concurrency optimizations using task-merging and data-partitioning forms of parallelism have been considered. This resulted in an optimized parallel model with the best computation and communication workload balance.

To evaluate the effectiveness of the optimized parallel model, the system-level Sesame/Artemis simulation framework (Coffland et al., 2003) has been used targeting multiple multiprocessor platforms (Krichene Zrida et al., 2010). It has been shown that the encoding performance obtained, in terms of frames per second, are getting linearly better with the number of simulated processors (assumed to be MIPS R3000) as presented in the figure 6.

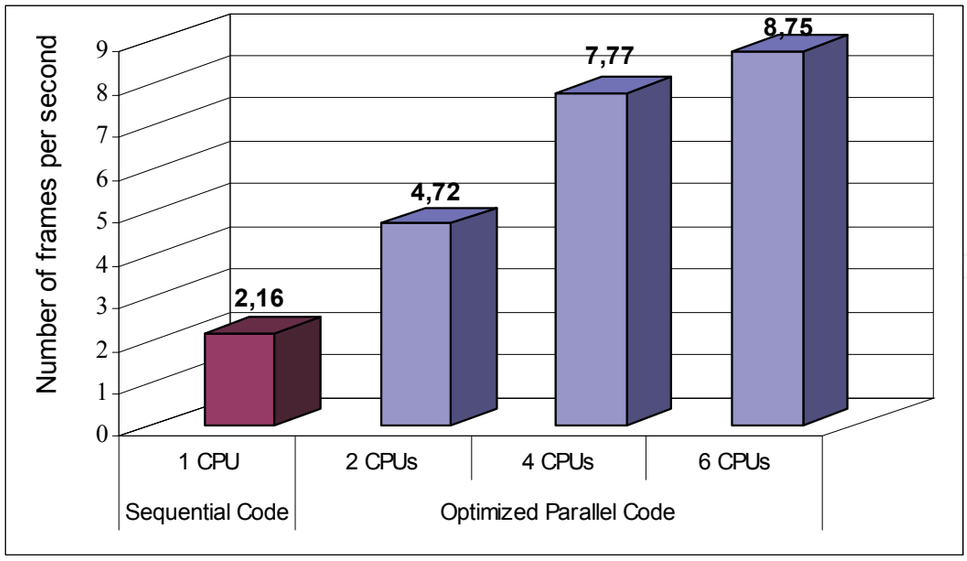


Fig. 6. Sesame/Artemis H.264 encoding performances vs. number of simulated processors

In addition, the encoding performance results of this optimized parallel model have also been compared to those previously obtained using the data-level parallelization approaches proposed in (Zhao, 2006; Sun, 2007). Results of this comparison are given in the table14. This table clearly shows that our solution (Krichene Zrida et al., 2009), based on simultaneous task and data level parallelism, has achieved better performance of the encoding process. Actually, using references (Zhao, 2006; Sun, 2007), data splitting is performed respectively at the Macro-Blocks MBs row and MBs region communication granularity levels. But for our case, a more fine-grain Macro-Block communication granularity level is exploited. Thus, with a more fine grain data amount exchanged by the processors, our proposed approach is more appropriate for use in embedded multiprocessor SoC implementations having limited on-chip memory resources.

	Number of processors	QCIF YUV frames Encoding simulation time (s)	Number of frames per second (fps)	Speedup	Speedup in (Zhao, 2006)	Speedup in (Sun, 2007)
Sequential H.264 code (JM10.2)	Mono-Processor	—	2.16	1	1	1
Optimized parallel H.264 model	2 Processors	1,6	4.72	2.19	—	—
	4 Processors	1.00	7.77	**3.6**	3.1	3.3

Table 14. Obtained Multiprocessor simulation results in comparison to those obtained in (Zhao, 2006; Sun, 2007)

Finally it has been shown, for a four-processor platform with the common bus structure, that the computation cost is much more important than the time spent in reading/writing from/to the shared memory. The communication and computation loads are nearly balanced for all the used components, as shown in the figure 7. These results represent again a solid confirm of the good concurrency properties of the obtained optimized model.

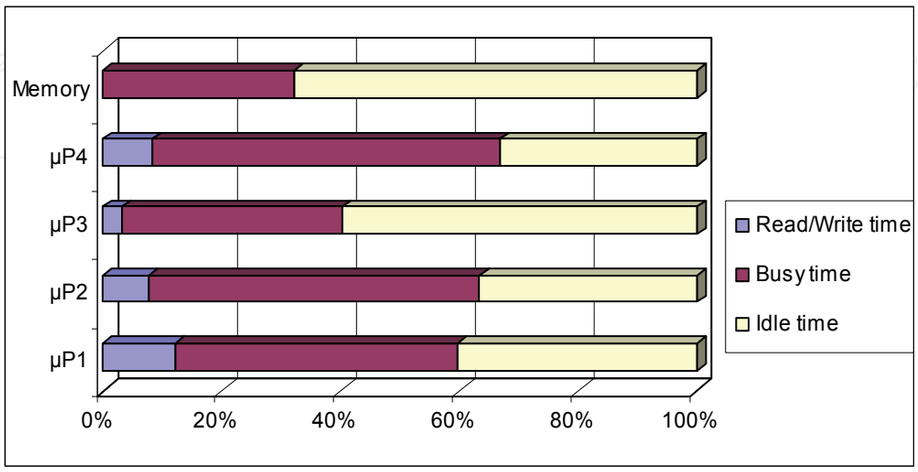


Fig. 7. Reading-Writing/Execution/Idle statistics for the common-bus-based architecture

However given the results of the figure 7, the times being idle are too much important in comparison with those being busy for all the architecture components. This has probably caused a substantial degradation of the final encoding performances. Given the important amount of data communicated between processes for the H.264/AVC encoding process, it is clear that the common memory bus structure constitutes a serious communication bottleneck. Actually, the very important data dependency between processors requires a potential memory access and allocation for the read/write operations. For a common-bus multiprocessor architecture, this causes a saturation of bus and thus a lot of time is spent in waiting to read/write data from/to other component. For further design space exploration and in order to reduce the communication bottleneck observed for the common-bus-based architecture, others inter processors communication structures and topologies need to be evaluated for a better encoding performance.

7. Conclusions

The H.264/AVC has been designed with the goal of enabling significantly improved compression performance relative to all existing video coding standards. Implementing a H.264 video encoder represents a big challenge for resource-constrained multimedia systems such as wireless devices or high-volume consumer electronics since this requires very high computational power to achieve real-time encoding. In this chapter, a high-level performance analysis of a H.264 video encoder is first performed to find an optimal balance between the coding efficiency and the implementation cost allowing for a complexity reduction at the high system level.

For an optimal use of the AVC tools, the best configuration parameters are obtained. For this cost-efficient configuration, the absolute complexity values, the memory and task level profiling results confirmed the big challenge needed for its effective implementation. For

such implementation, a multiprocessor approach is needed to share the encoding application execution time between several processors for achieving better execution performances and real time encoding.

8. References

Richardson, Iain E.G. (2003), H.264 and MPEG-4 *Video Compression: Video Coding for Next-generation Multimedia*, John Wiley & Sons Ltd.

ISO/IEC 13818 (1995), *Information technology: generic coding of moving pictures and associated audio information*, (MPEG-2).

Joch, A., Kossentini, F., & Nasiopoulos, P. (2002). A Performance Analysis of the ITU-T Draft H. 26L Video Coding Standard, *Proceedings of the 12th International Packet Video Workshop*, Pittsburg, Pa, USA, April 2002.

Alvarez, M., Salami, A., Ramirez, A., & Valero, M. (2005), A Performance Characterization of high Definition Digital Video Decoding using H264/AVC, *Proceedings of the IEEE International, Symposium on Workload Characterization*, pp. 24 – 33, 6-8 Oct. 2005.

Saponara, S., Denolf, K., Lafruit, G., Blanch, C. , & Bormans, J. (2004), Performance and Complexity Co-evaluation of the Advanced Video Coding Standard for Cost-Effective multimedia communication, *EURAPIS Journal on Applied Signal Processing*, pp. 220-235, 2004:2.

Kuhn, P. , & Stechele, W. (1998), Complexity analysis of the emerging MPEG-4 standard as a basis for VLSI implementation, *Proceedings of SPIE Visual Communications and Image Processing*, vol. 3309, pp. 498–509, San Jose, Calif, USA, January 1998.

Catthoor, F., Wuytack, S., De Greef, E., Balasa, F., Nachtergaele, L., & Vandecapelle, A. (2002), Data Access and Storage Management for Embedded Programmable Processors, *Kluwer Academic*, Boston, Mass, USA, 2002.

Chimienti, A., Fanucci, L., Locatelli, R., & Saponara, S. (2002), VLSI architecture for a low-power video codec system, *Microelectronics Journal*, vol. 33, no. 5-6, pp. 417–427, 2002.

Schäfer, R., Wiegand, T., Schwarz, H. (2003), The emerging H264/AVC standard, *EBU technical review*, January 2003.

Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., & Wedi, T. (2004), Video coding with H.264/AVC: Tools, Performance, and Complexity, *proceedings of the IEEE Circuits and Systems Magazine*, 2004.

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.

Malvar, H., Hallapuro, A., Karczewicz, M., & Kerofsky, L. (2003), Low-Complexity transform and quantization in H.264/AVC, *proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 598–603, July 2003.

H264 Reference Software Version JM 10.2
http://iphome.hhi.de/suehring/tml/. November 2005.

Arizona State University, Video traces for network performance evaluation,
http://trace.eas.asu.edu

Pascalis, P., Pezzoni, L., Mian, G.A., & Bagni, D. (2004), Fast Motion Estimation with size-based predictors' selection hexagon in H264/AVC Encoding, *proceedings of the 12th European Signal Processing Conference*, September 6-10 2004, Vienna, Austria

Chen, Z., & He, Y. (2002), Fast Integer and Fractional Pel Motion Estimation, JVT-E045, *5th Meeting: Geneva*, Switzerland, 9-17 October, 2002.

Denolf, K., Blanch, C., Lafruit, G., & Bormans, J. (2002), initial memory complexity analysis of the AVC codec, *proceedings* of the IEEE Workshop on Signal Processing Systems, pp. 222–227, San Diego, Calif, USA, October 2002.
www.gnome.org/projects/memprof/

Denolf, K. et al. (2000), Cost-efficient C-level design of an MPEG-4 video decoder, *proceedings of the IEEE Workshop on Power and Timing Modeling, optimization and Simulation*, pp. 233-242, Goettingen, Germany, September 2000.

Vleeschouwerand, C.D., & Nilsson, T. (2001), Motion estimation for low power video devicrs, *proceedings of the IEEE International Conference on Image Processing*, pp. 953-957, Creece, October 2001.

Kuhn, P. (1999), Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Kluwer Academic Publishers, 1999.

Graham, S.L., Kessler, P.B., McKusick, M.K. (1982), Gprof: A Call Graph Execution Profiler. *Proceedings of the SIGPLAN '82 Symposium on Compiler Construction*.
http://www.gnu.org/software/binutils/manual/gprof-2.9.1/

Li, P., Veeravalli, B., & Kassim, A. (2005), Design and Implementation of Parallel Video Encoding Strategies Using Divisible Load Analysis, *IEEE Transactions on Circuits and Systems for Video Technology*, No. 9, Vol. 15, pp. 1098-1112, September 2005.

Youssef, M., Yoo, S., Sasongko, A., Paviot, Y., & Jerraya, A.A. (2004), Debugging HW/SW interface for MPSOC: Video Encoder System Design Case Study, *proceedings of the 41st Design Automation Conference*, 2004.

Pastrnak, M., de With, P.H.N., Stuijk, S., & van Meerbergen, J. (2006), Parallel Implementation of Arbitrary-Shaped MPEG-4 Decoder for Multiprocessor Systems, *proceedings of the Visual Communications and Image Processing*, pp 60771I-1 - 60771I-10, 2006.

Gulati, A., & Campbell, G. (2005), Efficient mapping of the H.264 encoding algorithm onto multiprocessor DSPs. *proceedings of the SPIE-IS&T Electronic Imaging*, 2005

Chen, Y-K., Tian, X., Ge, S., & Girkar, M. (2004), Towards Efficient Multi-Level Threading of H.264 Encoder on Intel Hyper-Threading Architectures, *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004.

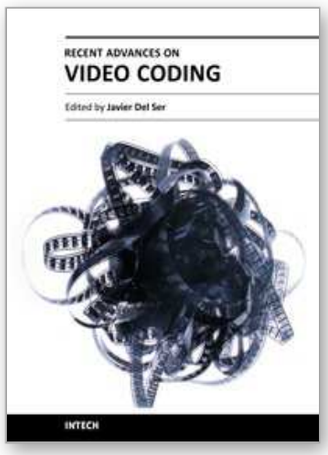
Zhao, Z., Liang, P. (2006), A Highly Efficient Parallel Algorithm for H.264 Video Encoder, *Proceedings of the 31st IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006.

Sun, Sh., Wang, D., & Chen, S. (2007), A Highly Efficient Parallel Algorithm for H.264 Encoder Based on Macro-Block Region Partition, HPCC 2007, LNCS 4782, pp. 577–585, Berlin Heidelberg 2007.

Krichene Zrida, H., Jemai, A., Ammari, A.C., & Abid, M. (2009), High Level H.264/AVC Video Encoder Parallelization for Multiprocessor Implementation, *Proceedings of the 12th ACM/IEEE Design Automation and Test in Europe conference and exhibition* , Nice-France, 20-24 April 2009.

Coffland. J.E., & Pimentel, A.D. (2003), A Software Framework for Efficient System level Performance Evaluation of Embedded Systems, *Proceedings of the SAC the ACM Symposium on Applied Computing*, Melbourne, Florida, USA, Mar. 2003.

Krichene Zrida, H., Ammari, A.C., Jemai, A., & Abid, M. (2010), High Level Optimized Parallel Specification of a H.264/AVC Video Encoder, *International Journal of Computing and Information Sciences (IJCIS)*, Volume 8, n°3, December 2010.



Recent Advances on Video Coding

Edited by Dr. Javier Del Ser Lorente

ISBN 978-953-307-181-7
Hard cover, 398 pages
Publisher InTech
Published online 24, June, 2011
Published in print edition June, 2011

This book is intended to attract the attention of practitioners and researchers from industry and academia interested in challenging paradigms of multimedia video coding, with an emphasis on recent technical developments, cross-disciplinary tools and implementations. Given its instructional purpose, the book also overviews recently published video coding standards such as H.264/AVC and SVC from a simulational standpoint. Novel rate control schemes and cross-disciplinary tools for the optimization of diverse aspects related to video coding are also addressed in detail, along with implementation architectures specially tailored for video processing and encoding. The book concludes by exposing new advances in semantic video coding. In summary: this book serves as a technically sounding start point for early-stage researchers and developers willing to join leading-edge research on video coding, processing and multimedia transmission.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen