Forward Wyner-Ziv Fast Video Decoding Using Multicore Processors

Alberto Corrales-Garcia[1], José Luis Martínez[2],
Gerardo Fernández-Escribano[1], and Francisco Jose Quiles[1]

[1] Instituto de Investigación en Informática de Albacete (I3A),
University of Castilla-La Mancha, Campus Universitario, 02071 Albacete, Spain
{albertocorrales,gerardo,paco}@dsi.uclm.es
[2] Architecture and Technology of Computing Systems Group. Complutense University
Ciudad Universitaria s/n, 28040 Madrid, Spain
joseluis.martinez@fdi.ucm.es

Abstract. With the aim of providing low complexity encoders, Wyner-Ziv video coding provides a new paradigm where the complexity of the encoder is moved to the decoder. However, this high decoding complexity could involve a problem in some applications which have delay restrictions. Nowadays parallel computing is a growing field into the computation market. In particular, most of personal computers and hardware for video coding includes multicore processors, which allows a parallel execution by means of several independent cores in a same chip. As a consequence, several DVC parallel decoding approaches are beginning to appear. This work proposes a parallel DVC decoding scheme for multicore processors, which decodes each GOP in an independent and parallel way. This scheme achieves above 70% time reduction without any rate-distortion penalty.

Keywords: Distributed Video Coding, Parallel Computing, Multicore Processors, OpenMP.

1 Introduction

Traditionally, the digital video codecs adopted by all MPEG and ITU-T video coding Standards have based their design in architectures where encoders are more complex than decoders [1]. Nowadays, new devices (such as surveillance systems, sensor networks, micro cameras, etc) with low-cost hardware can integrate cameras, but they should carry out a low-cost encoding. For this kind of applications, Wyner-Ziv (WZ) video coding [2] (which is a particular case of Distributed Video Coding (DVC) [3]) is an attractive paradigm since it provides a framework where the complexity of the encoder is displaced to the decoder allowing low cost encoding. This low-complexity is achieved because the encoding process does not exploit the temporal correlation to compress more the video information. In addition, DVC can achieve theoretically a similar Rate-Distortion (RD) performance than the joint video coding. At the same time, it provides robustness over noisy channel (as it often happens in wireless networks). Despite all this advantages, the complexity of the decoder is highly increased. Most of this complexity is caused by the iterative turbo decoding algorithm. In particular, the feedback channel contributes to a large degree in the cost of the decoder [4]. Although the amount of time taken by the decoder could not seem important, for many applications which the decoding delay plays an important role (such as WZ to H.264 video transcoding [3]), a fast decoding is desired and sometimes mandatory.

On the other hand, the technological advancement in microprocessors has introduced new architectures, which allow high-performance computing [5]. In particular, regarding processors, the new architectures tend to include several processors (called cores) in the same chip. This kind of processors is called Multicore Processors and nowadays they are widely extended in the market. However, although multicore processors can help to reduce the time spent by high-complex tasks, most of applications are designed to be executed in a sequential way. As a consequence, high complex tasks follow spreading much time and they do not take advance of the available computational capacity. To exploit this research field, the researching community should invest effort to propose new architectures and methods to take advance of the parallel computing to use efficiently the computational capacity that the new hardware offers to us.

At that point, this paper proposes to reduce the complexity of the WZ decoder (the WZ decoding complexity is even higher than traditional video coding algorithms [4]) by means of a multicore processor system. As a first attempt to achieve this, each Group of Pictures (GOP) is decoded in each processing unit. This provides good trade-off between the time reduction and the RD loss. The simulations results offer an acceptable time reduction up to 71% without any RD penalty. Moreover, the present proposal is scalable for a higher number of cores.

Accordingly, this paper is organized as follows: Section 2 presents an overview of the WZ codec and multicores; section 3 shows the previous works related to reducing WZ decoding complexity and some parallel DVC decoding approaches; section 4 proposes the parallel WZ decoder based on multicore; section 5 presents experimental results for the proposed architecture; and, finally, some final remarks are presented in Section 6.

2 Technical Background

2.1 Distributed Video Coding

Theoretical fundaments of DVC depart from the information theory [6]. However, is in [3] where one of the first practical DVC architecture was proposed by Stanford University. It is based on turbo codes as Slepian-Wolf encoder/decoder and a feedback channel used by the decoder to manage the rate control. Over the Stanford architecture many research and improvement have been carried out in the literature. Later, in [7] was proposed the DISCOVER codec architecture as a result of a European project and it could be considered as a reference codec in the DVC paradigm. In addition, DISCOVER codec was later improved by VINET-II team [8]. These codecs are focus on achieving the best RD results without considering the time spent. In this work, our architecture is an improved version of VISNET-II codec which is depicted in Figure 1.

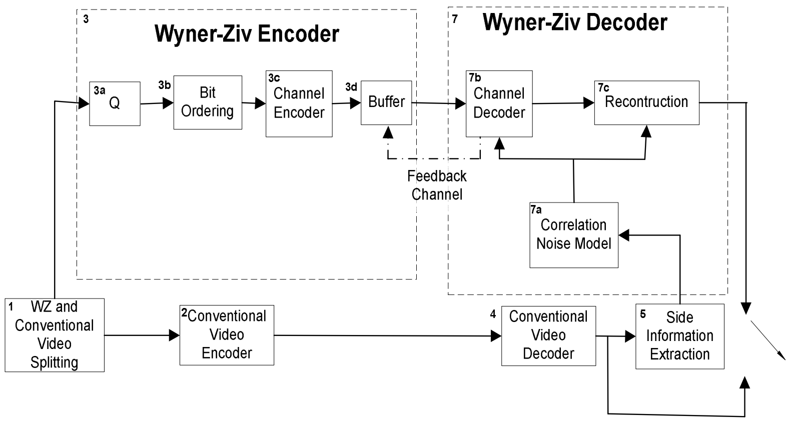
3 Wyner-Ziv Encoder
3a Q
3b Bit Ordering
3c Channel Encoder
3d Buffer
7 Wyner-Ziv Decoder
7b Channel Decoder
7c Recontruction
Feedback Channel
7a Correlation Noise Model
1 WZ and Conventional Video Splitting
2 Conventional Video Encoder
4 Conventional Video Decoder
5 Side Information Extraction

Fig. 1. Block diagram of the reference DVC architecture

The Figure 1 presents a scheme of the architecture employed in this paper. To sum up the basic WZ video coding architecture operation, we should know WZ video coding deals with two kinds of frames: Key Frames (K) and Wyner-Ziv Frames (WZ). Each frame of the sequence is sent to a different channel by means of the splitting module (1). At the encoder side, the K frames are encoded using a H.264/AVC Intra encoder [1] (2). On the other hand, the WZ frames are sent to a Wyner-Ziv Encoder (3). On the first stage, the frame information is quantized (3a). Afterwards, over the resulting quantized symbol stream, a bitplane extraction is performed per bitplanes (3b). Each bitplane is then independently channel encoded, starting with the most significant bitplane (3c). The parity bits produced by the channel encoder are stored in the buffer and transmitted in small amounts upon decoder request via the feedback channel; the systematic bits are discarded (3d).

On the other hand, in the decoder side, firstly the K frames are decoded using a H.264/AVC Intra decoder [1] (4). Then, in (5) the decoder uses each both frames like previous and next temporary references to create a Side Information (SI) frame. SI represents an estimation for each non-present original WZ frame. From each SI in (6) a Laplacian distribution models the residual statistics between corresponding WZ frame and SI. Then, the SI and the statistic model associated are used in an iterative decoding algorithm (7b) to obtain the decoder quantized symbol. In this module, each bitplane is decoded in a sequential order. Every decoding iteration new parity bits are requested to the encoder by means of the feedback channel. To decide if more parity bits are requested, a stopping criterion is defined based on error probabilities. When the decoding is considered successfully another bitplane is being decoded. Finally, the reconstructed pixels are obtained using the decoded quantized pixels, the correlation noise model estimated in (7a) and the quantized SI pixels.

2.2 Multicore Processor System

As a consequence of the computation limit of single processors, some time ago was introduced successfully the idea of having multiple cores in a same chip. Nowadays the usage of multicore processors is growing more and more. In fact, most of commercial computers include a multicore processor to increase the performance of the computers. In a muticore processor each core can execute a different application or they can work in a collaborative way to accelerate the execution of one application. However, due to heritage of simple processors, most of complex applications are designed to be executed in a sequential way and then the computational capacity of multicore systems are not fully exploited. In the particular case of multimedia applications, multicore processors can help to accelerate complex tasks. In fact, many multimedia hardware solutions are based on this kind of architectures. However, new methods and algorithms should be proposed to support the parallel execution as efficiently as possible.

In the architecture of a multicore processor, several cores share the same chip and they have some shared memory and some private memory. Regarding commercial multicore processors, the highest performance is reached by the multicore processors based on Intel Nehalem Micro-architecture [9]. In particular, this paper is based on this multicore processor Intel i7-940. The most important features of this processor are the following: four cores, clock speed of 2.93 GHz, 45nm manufacturing process, new point-to-point processor interconnect, Intel QuickPath Interconnect (QIP), Simultaneous Multi-Threading (SMT) by multiple cores which enables two threads per core (hyper-threading) and three levels of cache (32 KB L1 instruction and 32 KB L1 data cache per core, 256 KB L2 cache per core and 8 MB L3 cache shared by all cores).

On the other hand, Open Multi-Processing (OpemMP) has been proposed to develop parallel programs over this kind of multicore processors [10]. OpemMP is an Application Programming Interface (API) that supports multi-platform shared memory multiprocessing programming. It provides a portable and scalable model which consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

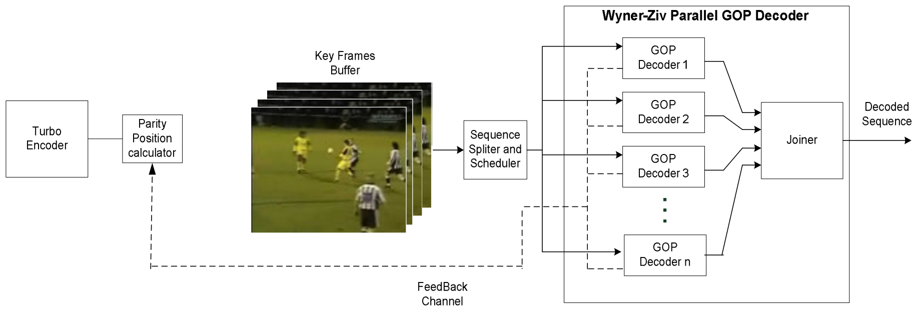

Fig. 2. Proposed Parallel DVC GOP decocing architecture

3 Related Work

DVC framework is based on displacing the complexity from the encoders to the decoders; however, a reduction of the complexity into the decoders is desirable. In traditional feedback-based DVC architectures [3], the rate control is done at the decoder and it is controlled by means of feedback channel; this is the main reason of the decoder complexity just because once a parity chunk arrives to the decoder, the turbo decoding algorithm (one of the most computational task [4]) is called. Taking this fact into account, there are several approaches which try to reduce de complexity of the decoder, which usually induces RD penalty. However, due to the technology advance, new parallel hardware is being introduced in practical video coding solutions. These new features of computers offer a new challenge for the research community to integrate its algorithms into the parallel framework; this opens a new door in the multimedia research. On the one hand, regarding the traditional standards several approaches have been proposed since multicores appeared in the market but, this paper focuses on parallel computing applied to the DVC framework.

On the other hand, in 2010 have been proposed different parallel solutions for DVC. In particular, in [11] *Oh et al.* proposed a DVC parallel execution carried out by Graphic Processing Units (GPUs). In this proposal, authors focus on design a parallel distribution for a Slepian-Wolf decoder based on rate Adaptative Low Density Check Code (LDPC) with Accumulator (LDPCA). LDPC codes are composed by many bit-nodes which do not have many dependencies between each node, so they propose a parallel execution in three kernels (steps): “kernels for check nodes calculations”, “kernels for bit nodes calculations”, and “kernels for termination condition calculations”. In a NVIDIA GeForce GTX260 216SP GPU they achieve a decoding 4~5 times faster for QCIF and 15~20 for CIF. On the other hand, in [12] *Momcilovic et al.* proposed a DVC LDPC parallel decoding based on multicore processors. In this work, the authors parallelize several LDPC approaches (Sum-Product, Min-Sum, and Algorithm E). In a Quad-Core machine, they reach and speedup up to 3.5. Both previous approaches propose a low level parallelism for a particular LDPC/LDPCA implementation.

However, the current work presents a higher level parallel WZ video decoding algorithm implemented over a multicore system. The reference WZ decoding algorithm is adapted by means of a GOP parallel decoding. In addition, the proposed algorithm is scalable because it does not depend on the hardware architecture neither the number of cores or on the implementation of the internal Wyner-Ziv decoder. Therefore, the time reduction can be increased simply by increasing the number of cores, as technology advance. Furthermore, the algorithm depicted in this paper could be extended by using another level of parallelism (frame or bitplane) as well as GPUs.

4 Proposed Wyner-Ziv GOP Parallel Decoding

Although most of commercial computers include multiple core processors, several cores are inactive regularly, while other are overloaded. As a result, the computation capacity is wasted and the complex tasks spend more time to finish. In the DVC framework, the fist aim is providing simple encoders, which are suitable to encode sequences in low cost devices. However, the decoder complexity is highly increased and sometimes, this high decoding delay does not allow including DVC in real environments. The first goal of this work is providing a simple and practical solution, which will be allow to execute WZ decoding in a parallel way, saving much decoding time and tanking advance of the computation capacity of whatever multicore processor.

4.1 Proposed Architecture

The traditional WZ decoding presents several sequential dependences such as the updating of CNM between bitplanes. Basically, once a bitplane is successfully decoded, the correlation model is updating by the following bitplanes.

Depending on the parallelism level, it is necessary to break these dependences, which could affect increasing the bitrate needed or decreasing the quality of the decoded frame. This is because a poor SI or a bad correlation model deals with a loss of performance. However, the WZ video coding offers an independent GOP decoding so we could execute each WZ GOP decoding in an independent core. In this way, we achieve a parallel decoder, which carries out a fast parallel decoding without any rate RD penalty. Figure 2 shows the proposed scheme, where the number of decoders is equal than the number of available cores, and then each decoder will decode one independent GOP. This involves that the architecture is not fixed for a specific hardware. In other words, it is architecture scalable.

In addition, the architecture has a module splitter and joiner. The first one carries out the task of splitting the key frame sequence by sending each key frame to the corresponding core. In addition, the joiner module includes the execution schedule. Each decoded GOP could have different delay due to the complexity of the scene and thus the number of the iterations needed. In addition, during a sequential execution there is one core in use and the rest are idle.

Taking this fact into account, the best schedule is a dynamic schedule (Figure 3), which assigns new tasks when any core finishes the current task. In this way, if there are GOPs to be decoded, all cores will be working and the capacity of the multicore processor will be utilized fully. In the end of the sequence, some cores could be idle, but this period is insignificant comparing with the whole sequence time decoding.

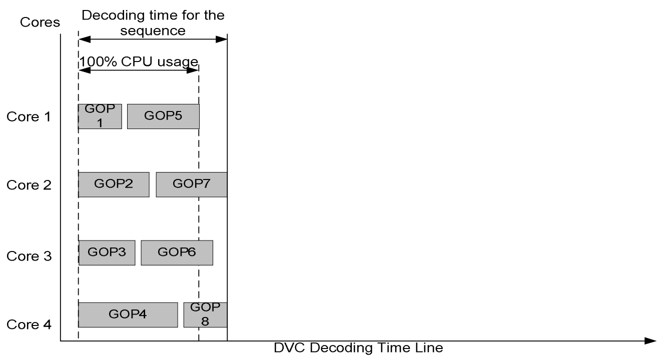

Fig. 3. DVC parallel GOP decoding time line execution for a 4 cores processor

On the other hand, the joiner module carries out the task of join each GOP in a suitable sequential way, because the parallel decoding could not maintain the source order.

In addition, as the decoding could be carried out without following a sequential order, the parity data could be also requested without a sequential order. To consider this case, the decoder sends a few bits in a header of the request to the encoder, which includes a module to estimate the relative position of the parity data related to each GOP. The *Parity Position* (*PP*) is calculated by the Equation 1, where *I* is the Intra period, *P* is position of the current GOP and *Q* is the quantification parameter. On the other hand *W* is the width of the image and *H* the height.

$$PP = (I - 1) * P * Q * \left(\left(\frac{W * H * 2}{8}\right) + 1\right) \tag{1}$$

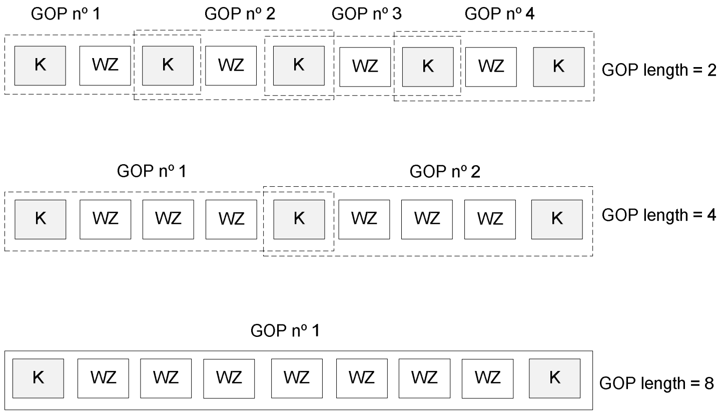

Fig. 4. Distribution of GOPs en each core and data shared for GOPs 2, 4 and 8

The WZ video coding only needs two reference frames to build any GOP length. Therefore, for consecutive frames, the same K frame is shared, as it is shown in Figure 4.Normally, in the WZ video coding the GOP sizes used are 2, 4 and 8.

In a nutshell, initially the parallel decoder needs to store several frames. This number of frames will depend on the number of available cores, but it does not depend on the GOP length. In general terms, the *Number of Frames* (*NF)* needed in the key frame buffer at the beginning is defined by the Equation 2, where *c* is the number of cores.

$$NF = \mathrm{c} + 1 \tag{2}$$

Finally, the structures created at the beginning for each core are reused for every GOP decoding, saving time and reusing the allocated memory.

5 Experimental Results

In order to evaluate the proposed parallel DVC decoding, four QCIF sequences were considered. These sequences have different motion and complexity features. For each sequence 150 frames were encoded by using the DVC VISNET-II codec [8]. The parallel implementation departs from the VISNET-II codec and was implemented by using Intel C++ compiler (version 11.1) [9] which combines a high-performance compiler as well as Intel Performance Libraries to support multi-threading applications. In addition, it provides support for OpenMP 3.0 [10]. In order to study the performance of the DVC parallel decoder, the sequences were encoded by using quantification from 1 to 4 bitplanes in pixel domain. In addition, to analyze the impact in different GOPs, several lengths of GOP were selected (2, 4 and 8).

The Time Reduction achieved (TR) is calculated by the Equation 3, where $Time_{seq}$ is the time spent by the sequential decoding and $Time_{par}$ the time spent by the proposed parallel version. Additionally, the speedup is calculated as the reference time divided by the parallel time.

$$TR = 100 * \frac{\left(Time_{seq} - Time_{par}\right)}{Time_{seq}} \tag{3}$$

The Table 1 shows the results for a GOP length = 2. It displays the results for several bitplanes (BPs) for each sequence. The Reference Time per frame column represents the decoding time spent by one WZ frame (on average). The reference version is composed by the VISNET-II sequential version. In the proposed parallel version a four-core processor was used (more details in section 2.2). As this multicore processor allows hyperthreading, each core runs 2 threads sharing the same core. In general, time reduction is higher in more complex sequences (such as foreman and soccer) reaching a mean of 71.05%. In most of the cases the speedup is between 3.5 and 4 (the maximum theoretically by using four core is 4). The RD results are not included due to for both versions (reference and proposed) are exactly the same.

On the other hand, tables 2 and 3 show the results for GOP length 4 and 8 respectively. As it is expected, the decoding time is a little higher for the middle frames because the distance of their references is higher and then, the SI generated is worse. To correct a worse initial SI, the decoding needs more interactions and then the decoding time per frame is increased. However, for longer GOP lengths similar conclusions are observed when sequential and parallel versions are compared.

In addition, a study about the influence of the number of cores and threads was done. Figure 5 shows the decoding time and the speedup factor (for Foreman sequence with GOP length = 2 and 3 BPs) when different threads are used in a 4 core processor with hyperthreading. As it is observed, the first 4 obtain a more significant time reduction whereas following 4 threads reach less time reduction. This is caused by the hyperthreading effect: when more than 4 threads are running, they are sharing the same physical cores and then the time reduction is lower.

Table 1. Parallel Decoder performance for GOP 2

Sequence	BP	Reference Time per frame (s)	Parallel Time per frame (s)	TR(%)	SpeedUp
Foreman	**1**	4.33	1.21	72.01	3.57
	2	7.49	1.94	74.12	3.86
	3	13.41	3.49	74.01	3.85
	4	20.05	5.39	73.13	3.72
Hall	**1**	3.18	1.21	62.07	2.64
	2	4.63	1.43	69.19	3.25
	3	8.35	2.18	73.88	3.83
	4	11.14	2.89	74.03	3.85
CoastGuard	**1**	3.05	1.22	59.95	2.50
	2	6.26	1.83	70.77	3.42
	3	11.97	3.37	71.82	3.55
	4	18.09	4.98	72.45	3.63
Soccer	**1**	7.18	2.07	71.21	3.47
	2	12.02	3.41	71.60	3.52
	3	19.47	5.24	73.06	3.71
	4	26.04	6.91	73.46	3.77
Mean		**11.04**	**3.05**	**71.05**	**3.51**

Table 2. Parallel Decoder performance for GOP 4

Sequence	BP	Reference Time per frame (s)	Parallel Time per frame (s)	TR(%)	SpeedUp
Foreman	**1**	4.33	1.21	72.01	3.57
	2	7.49	1.94	74.12	3.86
	3	13.41	3.49	74.01	3.85
	4	20.05	5.39	73.13	3.72
Hall	**1**	3.18	1.21	62.07	2.64
	2	4.63	1.43	69.19	3.25
	3	8.35	2.18	73.88	3.83
	4	11.14	2.89	74.03	3.85
CoastGuard	**1**	3.05	1.22	59.95	2.50
	2	6.26	1.83	70.77	3.42
	3	11.97	3.37	71.82	3.55
	4	18.09	4.98	72.45	3.63
Soccer	**1**	7.18	2.07	71.21	3.47
	2	12.02	3.41	71.60	3.52
	3	19.47	5.24	73.06	3.71
	4	26.04	6.91	73.46	3.77
Mean		**11.04**	**3.05**	**71.05**	**3.51**

Table 3. Parallel Decoder performance for GOP 8

Sequence	BP	Reference Time per frame (s)	Parallel Time per frame (s)	TR(%)	SpeedUp
Foreman	**1**	5.89	1.64	72.24	3.60
	2	10.71	2.65	75.26	4.04
	3	17.94	4.32	75.94	4.16
	4	25.72	6.40	75.10	4.02
Hall	**1**	2.81	1.19	57.6	2.36
	2	4.10	1.53	62.75	2.68
	3	6.75	2.12	68.66	3.19
	4	10.54	2.99	71.67	3.53
CoastGuard	**1**	3.61	1.41	60.86	2.56
	2	7.42	2.43	67.25	3.05
	3	13.36	4.11	69.27	3.25
	4	20.15	6.42	68.14	3.14
Soccer	**1**	8.89	2.26	74.51	3.92
	2	15.20	3.75	75.36	4.06
	3	23.12	5.72	75.26	4.04
	4	31.16	7.70	75.28	4.05
Mean		**12.96**	**3.54**	**70.32**	**3.48**

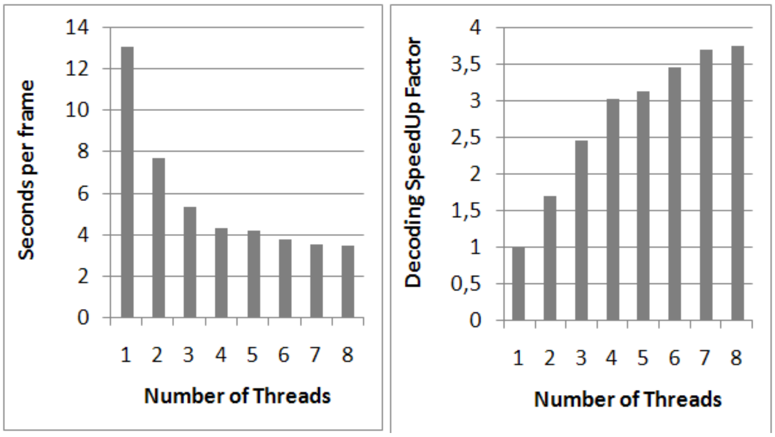

Fig. 5. DVC sequential decoding time line execution

6 Conclusions

The WZ video decoding is highly complex and this could be a problem for applications which have delay requirements. This work presents a WZ parallel decoding scheme by means of muticore processors. In this approach each GOP is decoding in an independent and parallel way, so that this scheme could be used in different WZ implementations without taking into account the implementations details of a particular approach. In addition, our proposed decoder maintains the same RD results than the sequential version. The time reduction reached is above 70% on average and it is extensible for longer GOP lengths with similar results. In spite of the feedback channel is still a bottleneck in DVC decoding, this proposed scheme could be used without modifications with more core architectures (following the current market tendency) and even in architectures without feedback channel.

Acknowledgments. This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grants CSD2006-00046, TIN2009-14475-C04. It was also supported by JCCM funds under grant PEII09-0037-2328 and PII2I09-0045-9916, and the University of Castilla-La Mancha under Project AT20101802. The work presented was performed by using the VISNET2-WZ-IST software developed in the framework of the VISNET II project.

References

1. ISO/IEC International Standard 14496-10:2003: Information Technology – Coding of Audio – Visual Objects – Part 10: Advanced Video Coding
2. Aaron, A., Rui, Z., Girod, B.: Wyner-Ziv coding of motion video. In: Asilomar Conference on Signals, Systems and Computers, pp. 240–244 (2002)
3. Girod, B., Aaron, A.M., Rane, S., Rebollo-Monedero, D.: Distributed Video Coding. Proceedings of the IEEE 93, 71–83 (2005)
4. Brites, C., Ascenso, J., Quintas Pedro, J., Pereira, F.: Evaluating a feedback channel based transform domain Wyner-Ziv video codec. Signal Processing: Image Communication 23, 269–297 (2008)
5. Feng, W.-C., Manocha, D.: High-performance computing using accelerators. Parallel Computing 33, 645–647 (2007)
6. Wyner, A.: Recent Results in the Shannon Theory. IEEE Trans. on Information Theory 20 (1974)
7. Artigas, X., Ascenso, J., Dalai, M., Klomp, S., Kubasov, D., Ouaret, M.: The DISCOVER codec: architecture, techniques and evaluation. In: Picture Coding Symposium (PCS), pp. 1-4. Citeseer (2007)
8. Ascenso, J., Brites, C., Dufaux, F., Fernando, A., Ebrahimi, T., Pereira, F., Tubaro, S.: The VISNET II DVC Codec: Architecture, Tools and Performance. In: European Signal Processing Conference, EUSIPCO (2010)
9. Intel Processor Core family, `http://www.intel.com/`
10. The OpenMP API specification for parallel programming, `http://openmp.org`
11. Ryanggeun, O., Jongbin, P., Byeungwoo, J.: Fast implementation of Wyner-Ziv Video codec using GPGPU. In: IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1–5 (2010)
12. Momcilovic, S., Yige, W., Rane, S., Vetro, A.: Toward realtime side information decoding on multi-core processors. In: IEEE International Workshop on Multimedia Signal Processing (MMSP), pp. 321–326 (2010)