Multi-Core Platform for DTV/STB Functional Testing in Real-Time

Dusan Milosavljevic and Vladimir Zlokolica

Abstract—**This paper presents a platform for DTV/STB functional testing, based on detection and measurement of video artifacts that originate from packet loss errors in transport stream. The system is capable for real-time processing of the DTV/STB video stream utilizing processing power of modern multi-core platform.**

Index Terms—**Packet-loss error detection, software parallelization, video quality assessment, video streaming.**

I. Introduction

ALONG with the development of multi-core platforms for personal computers, the past decade was marked by digitalization of television. Digitalization of television signal transmission was first performed in satellite transmission (DVB-S), and later the process of digitalizing was spread to terrestrial transmission, which is still carried out (DVB-T) in many countries. The choice of coding video signals depends on the available bandwidth. MPEG standard has been accepted for digital broadcasting of television signal as the basic and most commonly used for transmission of standard definition (SD) video content, whose basic concept is a digital packetized stream. By digital video transmission, it is obtained higher bandwidth utilization of transmission medium, and solved the problem of partial admission of broadcasted video signal, which is one of the basic conditions for guaranteeing the quality of service. However, the transmission errors that occur and which cannot be removed in the process of decoding by variety of methods may lead to packet losses in the transport stream. Such errors undermine the subjective quality of video content that in digital broadcasting has become very relevant for providing QoS [5], [6], [19], [22]-[24]. The new video services such as IPTV, Mobile TV and Internet Video Streaming rely on digital video transmission over networks, where the effects of packet-losses, due to various errors, are more frequent and more pronounced on the quality of video content [2].

In order to provide high quality experience of broadcasted video content to the end user, DTV and STB (set-top box) receivers apply error-concealment or complete discarding of poor quality video content based on decision from quality assessment as common approach. The correct and reliable functionality of DTV error concealment and error correction features is of great importance to the quality of video content that is experienced by the end user. In order to verify the required functionality of DTV devices, it is necessary to be tested in a realistic environment [25], [24] where device response on digital signal transmission errors is examined. The main functionality of the DTV/STB receiver, in addition to demodulation and signal decoding, is to provide the minimum required quality level of video content that is presented for certain error rate level in transport stream. Based on available and undamaged information extracted from bit-stream, the DTV/STB receiver tries to conceal errors manifested as visual degradation on video content by reconstructing missing information from video stream. Successful error concealment is directly related to broadcast video content and the number of lost packets. Consequently, based on the degree of visual degradation in video frames receiver determines whether to display decoded content of video stream over which is previously applied error concealment techniques. If the visual degradation of video frame is unacceptable due to high amount of present artifacts, the receiver should reject current video content and abort presentation of video frame on output. In this case from DTV is expected to show black screen with "no signal" message displayed or frozen video until better video quality is obtained.

This paper presents the implementation of DTV testing platform based on software solution algorithm for the detection of lost packets (PLD - Packet Loss Detection) [1] which gives a measure of errors during transport on the basis of artifacts in the image that impair visual quality of video content, primarily for video streams that are broadcasted over network. The software implementation of PLD algorithm is part of platform for testing TV and STB devices that provide a measure of video degradation that originate from packet loss errors in transmission. The emphasis is on the realization of such complex algorithm to explore the use of modern Intel multi-core architecture in the digital video signal processing. The main goal is to achieve execution times required for real time processing of video stream (by) applying the principles of

This work was partially supported by the Ministry of Education and Science of the Republic Serbia under the project No. 44009, year 2010. Part of results in this paper is presented on 55th ETRAN conference, Banja Vrucica, 6-9 June 2011.

D. Milosavljevic is with the Faculty of Technical Science, University of Novi Sad (e-mail: dusan.milosavljevic@rt-rk.com).

V. Zlokolica is with the Faculty of Technical Science, University of Novi Sad (e-mail: vladimir.zlokolica@rt-rk.com).

parallel programming and optimization techniques, that imply utilizing vector instructions (SSE, AVX) and OpenMP [26] standard for parallelization.

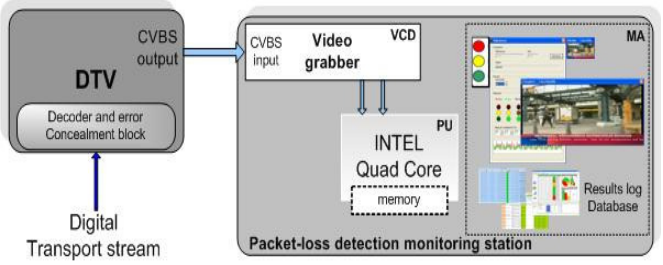

Fig. 1. Detailed description of the proposed system for the DTV functional testing.

II. GENERAL PLATFORM OVERVIEW FOR DTV TESTING

A. System description

In Fig. 1 the testing platform for DTV/STB devices is shown as separate testing station. The packet-loss detection monitoring station is a portable and modular Windows based embedded PC system, which is intended to be used by the DTV set and STB manufacturers or R&D, production and field device testing for packet-loss error issues (measurement and detection). The base of the platform is an Intel multi-core platform of the second generation (codenamed Sandy Bridge) where the software solution of PLD algorithm is being executed. The main idea of the platform for testing DTV and STB receivers is to the record media raw video directly from receiver by HDMI or CVBS outputs, which is further processed in real time by PLD algorithm to determine amount of artifacts that originate from packet loss errors. If the amount of artifacts is found to be above a predefined threshold, which is directly proportional to subjective visual perception, the DTV under investigation is acknowledged as dysfunctional. This platform consists of the three main components: (i) Video capturing device (VCD); (ii) Processing Unit (PU); (iii) Monitoring application (MA). The DTV under investigation is excited by digital transport stream, after which decoder of DTV/STB performs demodulation and consequently applies implemented error concealment techniques. The input of VCD is raw video signal extracted directly from DTV external sources such as CVBS or HDMI outputs. The captured raw video data is further streamed to The PU block, where the packet-loss detection is performed in real-time on INTEL multi-core platform. The PU provides the output measures for packet-loss errors corresponding to each frame and to the particular video sequence segment (VSS), which in our application is set to 50 frames (equals approximately 1s duration period). The latter is used to signal at the system output the packet-loss occurrence with three output states: (i) no error (labeled as green); (ii) undetermined/questionable error detection (labeled as yellow); (iii) certain error detection (labeled as red). These semaphore states are also shown in Fig. 1. The MA blog is responsible for managing test processes and determining states of test outcomes. Beside main functions and graphically displaying the test result the MA block is used to store testing results in database for later statistical analysis and debugging.

B. Use Cases

Beside primary functions such as inspecting the functionality of DTB devices there is another suitable use-case of presented platform. The use cases differ from each other in context environment in which same platform is used. The primary testing function is present on Fig. 2 where behavior of device under investigation is compared to referent device which functionality is proven in terms of error concealment specification. Furthermore, testing outcome can be conducted using non referenced method, when obtained PLM is investigated in respect to expected threshold specification in formed database. The other functionality of platform implies video quality monitoring measuring packet loss error rate of broadcasted stream. This configuration can be easily set if device under investigation is replaced with proven device. Video quality monitoring provided by this platform is very suitable for inspecting current error rate in broadcasted video streams that may be at level where error concealment methods become useless to decrease impact on video quality. Since packet losses is proportional to bit error rate, in noisy environment PLM measure could be useful to indicate low SNR of broadcasted signal.

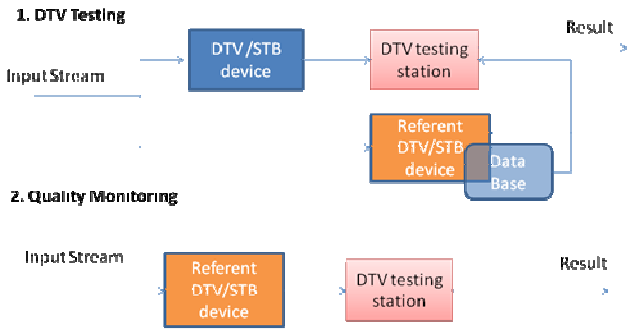

Fig. 2. System use cases for different set up environment: (top) – primary platform function for DTV testing; (bottom) – quality monitoring function.

III. PROPOSED PACKET LOSS ERROR DETCETION ALGHORITHM

In an MPEG-2 based video stream, data loss reduces quality depending strongly on the type of the lost information. Losses of syntactic data, such as headers and system information, affect the quality differently than losses of semantic data such as pure video information (e.g. motion vectors, DCT coefficients, etc.). Furthermore, the quality reduction depends on the location of the lost semantic data due, not only to the predictive structure of MPEG-2 video coded streams, but also to the visual relevance of the data [3].

The packet loss errors can affect different information and they manifest as visual blocking artifacts with block-wise

shapes, of different sizes, textures and colors. In Fig. 3, example of degraded video frame is shown.

In the past and recently, a considerable number of papers have addressed the topic of packet-loss error modeling [5]-[8], detection and monitoring [11]-[20] and its concealment [9], [10] in various video streaming applications such as broadcast [12], mobile [18], [20] and IP [19], [21], [22]. In [7], [17], [18] authors propose a parametric model for quality assessment based on network-level measurements and processed data in the transport stream. Alternatively, another approach for packet-loss video quality monitoring has been investigated in [11]-[15], where solely decoded and error concealed video frames are used for estimating packet-loss error. The latter approach is much more difficult task but is useful in case when only raw video data is available as it is the case for the end-user applications such as mobile devices and TV. Additionally, a hybrid approach where both the encoded and decoded video information is used for packet-loss detection was reported in [16].

The proposed packet-loss measurement algorithm is based on processing the decoded video stream, i.e., only the raw image information, where the packet-loss measure for the VSS is determined as a square root of the weighted square sum of the detected artifacts in each frame separately(PLM_i):

$$PLMS = \sqrt{\sum_i w_i PLM_i} \qquad (1)$$

Specifically, only the artifact measures (from each frame separately) that are found to be above a predefined threshold TA are included in the sum. This is regulated by weight factors w_i which values are 1 if PLM is higher than Ta. The algorithm is based on detecting blocking artifacts identifying the significant "sharp" horizontal edges that blocking artifacts consist of. Based on the vertical gradients, sharp horizontal edges (for both luminance and chrominance components), with low gradient activity in the local neighborhood, are determined. The algorithm procedure is divided on three separate sequential computations:

1) Gradient Evaluation (GE)
2) Gradient Filtering (GF)
3) Gradient Grouping and Summations (GGS)

A. Gradient Evaluation (GE)

First step of PLD algorithm is computation of vertical gradients applying the high pass filter with coefficients [-1.1]. The horizontal luminance gradients are computed as $G_{YH}(m,n)=|Y(m,n)-Y(m-1,n)|$, the vertical luminance gradients as $G_{YV}(m,n)=|Y(m,n)-Y(m,n-1)|$ and the chrominance vertical gradients as $G_{CV}(m,n)=(|U(m,n)-U(m,n-1)|+|V(m,n)-V(m,n-1)|)/2$, where *Y, U* and *V* refer to the luminance and chrominance components, and *m* and *n* are horizontal and vertical coordinates, respectively. The vertical gradients are further used to determine steep (sharp) horizontal edges with accentuated low gradient activity in the local neighborhood perpendicular to the edge direction. Left and/or right side of the edge are observed, based on which two types of sharp (significant) edges are determined (2): (i) left sharp edge (E_L) and (ii) right sharp edge (E_R);

$$E_l^{(H)} = K_1 G_v(m,n) - K_2 (K_3 G_v(m,n-1) + G_v(m,n-2) + G_v(m,n+1)) \qquad (2)$$

$$E_r^{(H)} = K_1 G_v(m,n) - K_2 (K_3 G_v(m,n+1) + G_v(m,n+2) + G_v(m,n-1)) \qquad (3)$$

where K1, K2 and K3 are parameter constants used for fine tuning and G stands for the luminance and chrominance vertical gradients, as previously defined.

Fig. 3. Video frame degraded by packet-loss.

B. Gradient Filtering

After the sharp horizontal edges (corresponding to vertical gradients) are computed for the color and luminance components, it is applied low-pass filtering along the edge direction using 5-tap filter in order to reject spurious edges from further calculation. On both the luminance and chrominance components the following filtering is performed.

C. Gradient Grouping and Summations

After the sharp horizontal edges have been determined for the luminance and chrominance components, they are grouped separately within the overlapping blocks and summed in a particular manner where only the edge filtered (EF) values that are larger than a predefined threshold (T_E) are considered and included in the summation. In case of the color components only the vertical gradients are utilized,, sharp horizontal edges, and summed separately for the "left" (SEF_{CHL}) and "right" (SEF_{CHR}) edges. Based on that the final blocking measure from color components for the whole frame is determined as $CM = SEF_{CHL} + SEF_{CHR}$, where the overlapping block is marked as "artifact affected" due to packet-loss if CM is larger than a predefined threshold T_{CHA} .

For the luminance components we perform more complex horizontal gradient and horizontal edge analysis, in order to determine the amount of the degradation since the luminance component carries more important information than the chrominance components. In case of luminance components the overlapping block (size of B_XxB_Y as 32x24 in our implementation) is divided in B_Y lines and for each line the number of connected horizontal edge pixels related to either "left" or "right" edges is calculated. The existence of the "edge line artifact" is acknowledged if the any of the sum (CN) is larger than a predefined threshold TCN. Based on this, we consider two cases: (i) single artifact lines (TCN = B_X/2)

and (ii) paired artifact lines (TCN=B_X/4). In case (ii), only the paired lines with vertical gap of 4, 8 and 16 lines are taken into account for artifact calculation since the block wise artifacts are in most cases assumed to be of 4 lines width and the slice and macro-block wise artifacts are assumed to be of either 8, 16 or more lines width.

D. Final measure

The final artifact measure PLM_i, for a video frame *i*, due to packet-loss is determined as a sum of detected the ”artifact affected” overlapping blocks, that come from the color horizontal sharp edges and the luminance horizontal sharp edges (single and multiply paired).

IV. ALGORITHM IMPLEMENTATION WITH SOFTWARE OPTIMIZATIONS AND PARALLELIZATION

In this section it is generally described optimizations and parallelization strategies used in implementation.

Since the target platform is multi-core Intel platform, in the implementation of PLD algorithm the vector instructions from SSE and AVX instruction set are used. Vector instructions are particularly suitable for video processing and image processing, where the on different data are applied the same operation (SIMD - Single Instruction Multiple Data). The new platform features new AVX instruction set, which provides twice as wide operands in the vector instructions in comparison to the SSE instruction set. Therefore, it enables two times higher level of parallelization for floating point arithmetic. AVX instruction set includes only arithmetic operations on vector floating-point types.

A. Vectorization

The AVX instruction set is utilised in a special version of the software solution only in the last stage of processing, which requires the calculation upon floating point values. When calculating vertical gradients (GE block) a vector of 16 pixels with a single-byte values from two successive rows of the both YUV components is loaded. Then, because of possible overflow in calculations of absolute difference of two vectors due to SSE instructions limitations. each vector must be unpacked into two vectors of two-byte pixel values . After subtraction and operation of the absolute values are performed, the two result vectors are obtained that are further packed into one vector with 16 values of the calculated vertical gradient.

By calculating the gradients in one iteration it is produced16 gradient values for the respective position of the input vectors. The two resulting vectors of 8 short integer words are repacked into a single vector of 16 bytes. Repacking seems as wasteful operations, because in the next step for sharp edges computation type conversion of values is performed. However, a vector of 16 bytes can be loaded in one memory transaction, thereby saving memory bandwidth by favoring execution of multiple instruction instead of multiple memory access. The calculation of sharp edges requires transition from integer arithmetic to floating point arithmetic. At this stage initially calculated gradients has to be converted from “char” to: “float” type, what causes a change in the organization of data within the vector, as well as reducing the parallelization degree of vector instructions. Hence, from one 16 byte vector four values are obtained and incorporated in parallel for arithmetic operations related to the “sharp” edge calculation. Preparation of vector processing includes unpacking 16 byte vectors and creating vectors of four 32 bit float values.

In computation of sharp edge gradients, for each vector of four from previous phase (4 vectors of floats produced from vector of 16 bytes) is necessary to supply three vectors from adjacent rows in frame with gradient values, in accordance with relations (2) and (3).

Block for filtering the significant sharp edges (GF) is implemented as a one-dimensional convolution using corresponding SSE instructions

Filtered values of characteristic edges are grouped into overlapping blocks of size 32x24 pixels in the final stage of processing (GGS) in which they are further summarized after they are previously compared with the vector of a certain threshold value. Overlapping blocks are moved by 4 elements in the horizontal or vertical directions within video frame with corresponding filtered sharp edge values. Block moving by 4 elements (float values) is particularly suitable because it provides a memory-aligned read of vector within overlapping block that has to be aligned to 16 bytes in case of using SSE instructions. Aligned memory read to the size of vector is much faster than unaligned read, since it spends only one memory access to transfer the entire vector of data at once [27].

To reduce the number of instructions to calculate the partial sums in overlapping block, and save valuable memory bandwidth when overlapping block is moved horizontally or vertically, it is applied re-using of the partial sum values of the neighbouring overlapping block from the previous position.

B. Multi-core Parallelization

Special attention in the process of optimizations and parallelization is taken on work scheduling and work balancing for algorithm processing stages across the processor cores. Parallel processing on multi cores is achieved using OpenMP [28] parallel application programming interface, which is defined as extension for C programming language.

Parallelization to multiple cores is based on the SPMD [29], [30] (Single Program Multiple Data) known methodology that corresponds to data-parallel model of the data domain decomposition paradigm. There are two implemented variants of solutions that use the decomposition of the data in two different domains.

The first method involves the data decomposition in the spatial domain, where the work from all processing stages of algorithm is distributed by geometric data partitioning from one video frame on horizontal parts that is intended to be processed in parallel on each core. The frame is divided vertically in horizontal continuous sections as many as there are processor cores available. The second method involves the data decomposition in the time domain, where consecutive frames from video sequence are assigned to each core for

processing. Number of video frames that are processed at the same time is determined by the number of cores.

By the appropriate OpenMP constructs, static scheduling of work-load distribution among program threads is chosen to minimizes the time required for scheduling and invoking program threads that are being executed on associated processor cores. The applied data partitioning in processing provides balanced workload for threads, therefore any possible mutual waiting is avoided. The only interaction between the processing threads is the reduction of private sums that contain number of detected artifacts blocks in each section or group assigned to particular thread in computation. At that point the degradation levels, computed on each core separately, have to be summed to provide the final packet-loss detection measure.

V. Experimental Results

The results for the proposed algorithm for measurement of the visual artifacts due to packet-loss were evaluated on several SD video sequences in progressive and interlace format; a group of interlace sequences considered belong to one packetized transport stream and the sequences in progressive format belong to the other considered transport stream. In the following we first present results concerning the algorithm sensitivities to artifacts and image content, as well as the robustness and accuracy of the proposed quality monitoring and detection system. After that we show the performance results in terms of the processing time per frame achieved by optimizations in context of vectorization with SSE and AVX instruction sets, and multi core parallelization.

A. *Metric verification results*

In order to test the performance of the proposed packet-loss measurement algorithm, we have first applied it on simulated attenuated signal (as it is the case in transmission), which is an input transport stream (progressive format) degraded (by software means) by 6 levels, where the first degradation level corresponds to non-degraded video sequences. In the second degradation level, in every 100.000 bits one bit was randomly set to zero. Additionally, in degradation level 3, two bits were degraded, in level 4 four bits, in level 5 eight bits and in level 6 sixteen bits were degraded. In order to obtain reliable results, we have made 3 series of the 6 degradation levels, where in each series different set of bits were degraded and thus different video artifacts were introduced to decoded video sequences. After that, the video stream was unpacked and decoded (the pre-trial decoder incorporated drops frozen frames) in order to acquire video content for display and analysis. Within the video stream three video sequences (length of approximately 500 frames) with different spatial and temporal context were considered and processed.

The results were also compared to the state-of-the-art algorithm [12]. In Fig. 4, the performance results for the three video sequences (in progressive format within one transport stream) in three degradation series with the 6 degradation levels are shown. From the figure, one can observe that the proposed PLMS measure is monotonically increasing function with the packet loss degradation level growth and is relatively robust against different video sequences in different degradation series. Although the results for sequences 3 in Fig. 5 are slightly different from those for the sequences 1 and 2, we argue that the proposed PLMS algorithm shows nevertheless good consistency. Namely, depending on the spatial and temporal context in different video sequences, the packet-loss error can be more or less apparent and more or less successful error concealment can be done within the decoding process. Consequently, small result variations for different sequences are acceptable and in practice inevitable.

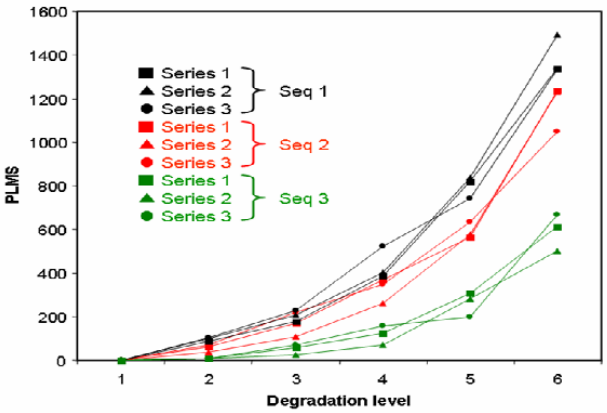

Fig. 4. The PLMS of three different SD video sequences in progressive format with 6 degradation levels and 3 series, where the first degradation level corresponds to the non-degraded video stream.

Additionally, we have compared the results of the proposed PLD algorithm to the algorithm for packet-loss impairment metric (PIM) of [12] (measure for the whole video sequences was also determined in the same manner as for the PLD algorithm), which is presented for one degradation series in Fig. 8. From the figure it is clear that the proposed method is superior in terms of the higher sensitivity to different degradation levels, while the robustness against different sequences is similar.

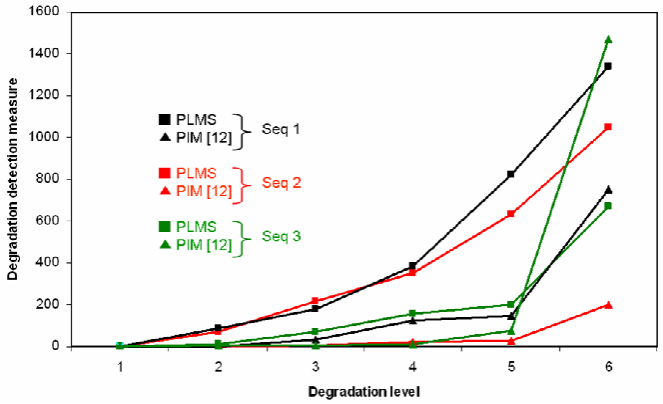

Fig. 5. A comparison between the proposed PLMS algorithm and PIM [12] on three different SD video sequences in progressive format with 6 degradation levels and 1 series, where the first degradation level corresponds to the non-degraded video stream.

B. *Processing performance results*

Performance results were obtained using standard video sequences SD of resolution 720 x 576 at 25 frames (per second in progressive video, that are randomly corrupted in manner as previously described. All measured performance results are expressed as execution given in milliseconds for processing per video frame. Performance measurements were conducted on last generation Intel i7 2600K with 4 GB of DDR3 RAM operating on 1600 MHz for two different clock processors frequencies.

The results for increased processor clock speeds have been obtained in order to investigate the ultimate limits of the system and get maximum possible performance results.

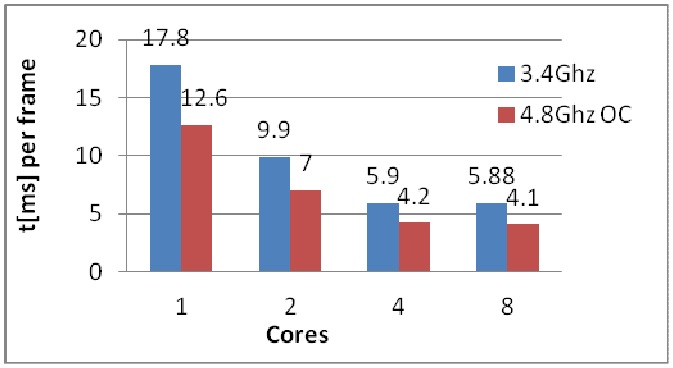

Fig. 6. Parallelization performance results achieved by SSE instruction set and spatial domain decomposition.

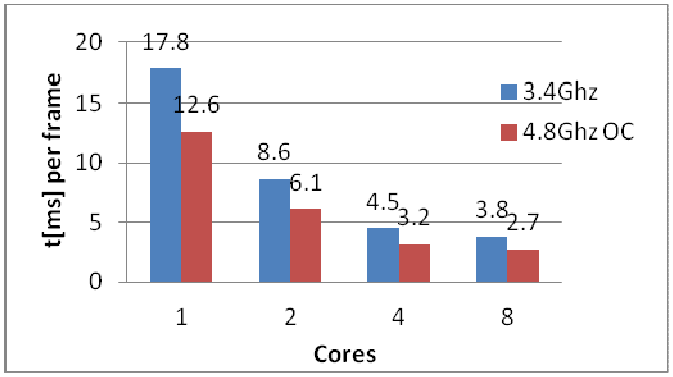

Fig. 7. Parallelization performance results achieved by SSE instruction set and time domain decomposition.

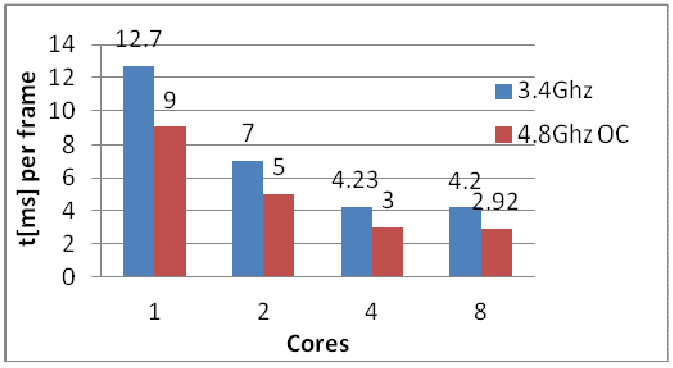

Fig. 8. Parallelization performance results achieved by AVX instruction set and spatial domain decomposition.

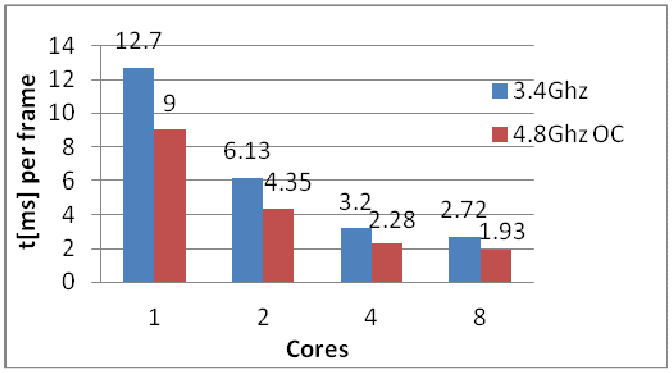

Fig. 9. Parallelization performance results achieved by AVX instruction set and time domain decomposition.

Execution time results of referent version of software solution considered as scalar naive implementation that is executed sequentially on one processor core for clock frequencies 3.4 and 4.8 GHz are respectively 73 and 52 ms per frame.

Important note for the results on 8 cores is that processor contains only four physical cores that can execute two independent stream of instructions from different program threads and expose up to 8 logical cores to applications. This technique is known as Hyper-Threading on Intel platforms. Targeted executions time for real time processing for a video stream with 25 frames per second is 40 ms per frame.

VI. Conclusion

In the implementation of the algorithm that is part of a platform for functional testing of DTV systems, video stream processing in real time is achieved on the Intel platform. The biggest limitation that affects the performance scalability on Intel multi-core platforms is the off-chip memory bandwidth that becomes the most precious and most expensive resource in terms of impact on overall performance. Although multi-core platform of the last generation relaxes this limitation through higher memory bandwidth, optimization techniques aimed at the rational use of this resource will likely provide a proportional scaling performance for parallel applications on future architectures with a much larger number of cores. Another important factor for increasing the performance of parallel applications is the efficient use of cache, which allows independent and simultaneous operation executing on processor cores without waiting for data and conflicting for exclusive access to shared memory. Implemented solution with applied optimization and efficient parallelization has processing capability of testing DTV devices with video streams of high definition resolution (HD).

References

[1] N. Teslic, V. Zlokolica, V. Pekovic, T. Tekcan, M. Temerinac, "Packet-Loss Error Detection system for DTV and set-top box functional testing", IEEE Transactions on CE, Volume 56, Issue 3, August 2010.

[2] O. Verscheure, P. Frossard, and M. Hamdi, "User-oriented QoS analysis in mpeg-2 video delivery," Real-Time Imaging, vol. 5, pp. 305–314, 1999.

[3] H.-C. Shyu and J.-J. Leou, "Detection and concealment of transmission errors in MPEG-2 images – genetic algorithm approach," IEEE

Transactions on Circuits and Systems for Video Technology, vol. 9, no. 6, pp. 937–948, September 1999.

[4] A. R. Reibman, D. Poole, “Characterizing packet loss impairments in compressed video,” in International Conference on Image Processing, San Antonio, Texas, USA, 2008, IEEE, vol. 5, pp. 77–88.

[5] Y. J. Liang, J. G. Apostolopoulus, B. Girod, “Analysis of Packet Loss for Compressed Video: Effect of Burst Losses and Correlation Between Error Frames,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 18, No. 7, July 2008, pp. 861-874.

[6] G. Haßlinger and O. Hohlfeld, ”The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet,” in 14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems, 2008, pp. 269-286.

[7] S. Kanumuri, P.C. Cosman, A.R. Reibman, and V.A. Vaishampayan, “Modeling Packet-Loss Visibility in MPEG-2 Video,” IEEE Transactions on Multimedia, vol. 8, no. 2, pp. 341–355, April 2006.

[8] T. Zhang, U. Jennehag, Y. Xu, “Numerical modeling of transmission errors and video quality of MPEG-2,” Signal Processing: *Image Communication* 16, 2001, 817-825.

[9] E. Asbun, E. J. Delp , "Real-time Error Concealment in Compressed Digital Video Streams, Picture Coding Symposium, Portland, OR, USA, 1999.

[10] S. Kaiser, K. Fazel, “Comparison of error concealment techniques for an MPEG-2 video decoder in terrestrial TV-broadcasting,” Signal Processing: Image Communication, Vol. 14 , 1999, pp. 655-676.

[11] R.V. Babu, A. S. Bopardikar, Perkis A., and O. I. Hillestad, “Noreference metrics for video streaming applications,” in 14th International Workshop on Packet Video, Irvine, California, USA, December 2004.

[12] H. Rui, C. Li, and S. Qiu, “Evaluation of packet loss impairment on streaming video,” Journal of Zhejiang University SCIENCE B, vol. 7, pp. 1– 7, 2006.

[13] N. Montard and P. Bretillon, “Objective quality monitoring issues in digital broadcasting networks,” IEEE Transactions on Broadcasting, vol. 51, no. 3, pp. 269–275, September 2005

[14] D. Shabtay, N. Raviv, and Y. Moshe, “Video packet loss concealment detection based on image content,” in 16th European Signal Processing Conference, , August 25-29, 2008,, Lausanne, Switzerland.

[15] E. P. Ong, S. Wu, M. H. Loke, S. Rahardja, J. Tay, C. K. Tan, L. Huang, “Video quality monitoring of streamed video,” in International Conference on Acoustics, Speech and Signal Processing, Taipei, Taiwan, 2009, IEEE, pp.1153-1156.

[16] T.Yamada, Y. Miyamoto, M. Serizawa, "No reference video quality estimation based on error concealment effectiveness," 16th International Packet Video Workshop, 2007, Lausanne, Switzerland.

[17] R. Reibman, V. A. Vaishampayan, and Y. Sermadevi, “Quality monitoring of video over a packet network,” IEEE Transactions on Multimedia, vol. 6, no. 2, pp. 327–334, April 2004.

[18] . Gustaffson, G. Heikkila, and M. Pettersson, “Measuring multimedia quality in mobile networks with an objective parametric model,” in International Conference on Image Processing, San Diego, USA, 2008, IEEE, vol. 5, pp. 405–408.

[19] S. Tao, J. G. Apostolopoulus, R. Guerin, “Real-Time Monitoring of Video Quality in IP Networks”, International Workshop on Network and Operating System Support for Digital Audio and Video, Stevenson, Washington, USA*, 2005, pp.* 129–134.

[20] T. Liu, G. Cash, W. Chen, C. Chen, J. Bloom, "Real-time Video Quality Monitoring for Mobile Devices," in Proceedings of the 44th Annual Conference on Information Sciences and Systems, Princeton, USA, 2010.

[21] D. Károly, T. Szemethy and Á. Bakay, “System and Signal Monitoring for IPTV Set-Top-Box Systems”. Regional Conference on Embedded and Ambient Systems RCEAS'07. November 2007. Budapest, Hungary.

[22] M. S. Siddiqui, S. O. Amin, C. S. Hong, “A Set-top Box for End-to-end QoS Management and Home Network Gateway in IMS,” IEEE Transactions on Consumer Electronics, Vol. 55, No. 2, May 2009, pp. 527-534.

[23] Y.-W. Suh, S.-H. Kim, M.-S. Kim, J.-Y. Choi, J.-S. Seo, “A Novel Integrated Measurement and Analysis System for Digital Broadcasting,” IEEE Transactions on Consumer Electronics, Vol. 55, No. 2, May 2009, pp. 56-62.

[24] S. D. Servetto, K. Nahrstedt, “Broadcast Quality Video over IP,” IEEE Transactions on Multimedia, Vol. 3, No. 1, March 2001.

[25] P.-C. Liu, Y.-D. Lin, W. J. Tsai, “Video Quality in Set-Top Box: Improving Test Productivity While Retaining Quality Index,” in Proc. International Computer Symposium, Taipei, Taiwan, Vol.3, pp.1185-1189, Dec., 2006.

[26] The OpenMP Architectures review Board ”OpenMP – The OpenMP API specifications for parallel programming.”, Available: http://www.openmp.org

[27] Shameem Akhter, Jason Roberts: *Multi-Core Programming, Increasing Softaware Performance through Software Multi-Threading*, Intel Press, 2006.

[28] Barbara Chapman, Gabriele Jost, Ruud van der Pas: *Using OpenMP Portable Shared Memory Parallel Programming*, MIT Press Cambridge, 2008.

[29] Tymothy G, Mattson, Beverly A. Sanders: *Patterns for Parallel Programming,* Software Patterns Series, 2004.

[30] Richard Gerber, *Software Optimization Cook-Book,* Intel Press 2002