Real-time SHVC Software Decoding with Multi-threaded Parallel Processing

Srinivas Gudumasu[a], Yuwen He[b], Yan Ye[b], Yong He[b], Eun-Seok Ryu[c], Jie Dong[b], Xiaoyu Xiu[b]
[a]Aricent Technologies, Okkiyam Thuraipakkam, Chennai, 600096 India; [b]InterDigital Communications, Inc., 9710 Scranton Road, San Diego, CA 92121, USA; [c]Mobile Communication Division, Samsung Electronics, 129, Samsung-ro, Yeongtong-gu, Suwon, 443-742, Korea

ABSTRACT

This paper proposes a parallel decoding framework for scalable HEVC (SHVC). Various optimization technologies are implemented on the basis of SHVC reference software SHM-2.0 to achieve real-time decoding speed for the two layer spatial scalability configuration. SHVC decoder complexity is analyzed with profiling information. The decoding process at each layer and the up-sampling process are designed in parallel and scheduled by a high level application task manager. Within each layer, multi-threaded decoding is applied to accelerate the layer decoding speed. Entropy decoding, reconstruction, and in-loop processing are pipeline designed with multiple threads based on groups of coding tree units (CTU). A group of CTUs is treated as a processing unit in each pipeline stage to achieve a better trade-off between parallelism and synchronization. Motion compensation, inverse quantization, and inverse transform modules are further optimized with SSE4 SIMD instructions. Simulations on a desktop with an Intel i7 processor 2600 running at 3.4 GHz show that the parallel SHVC software decoder is able to decode 1080p spatial 2x at up to 60 fps (frames per second) and 1080p spatial 1.5x at up to 50 fps for those bitstreams generated with SHVC common test conditions in the JCT-VC standardization group. The decoding performance at various bitrates with different optimization technologies and different numbers of threads are compared in terms of decoding speed and resource usage, including processor and memory.

Keywords: HEVC, scalable video coding, parallel processing, real-time decoding, SIMD

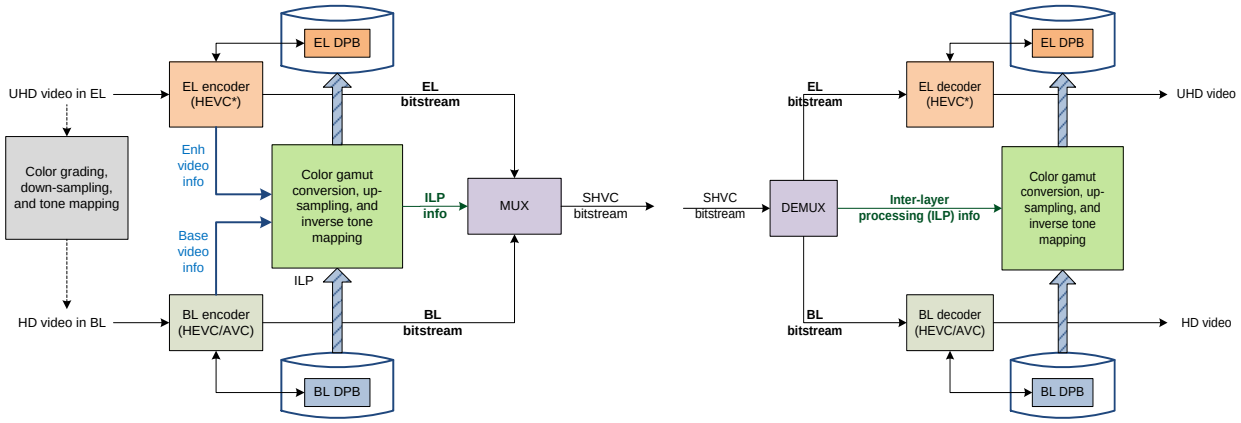
1. INTRODUCTION

Until recent years, digital video services primarily referred to TV services over satellite, cable and/or terrestrial broadcasting channels. However, as the internet on mobile devices starts to take root in people's daily lives, especially with the recent explosive growth of smartphones and tablets, both in resolution and computation capability, more and more new video applications, such as video chat, mobile video recording and sharing, and video streaming, require video transmission in heterogeneous environments [4]. The scenarios known as 3-screen and N-screen consider accommodating video consumption on various consumer devices (PCs, smartphones, tablets, TVs) with widely varying capabilities in terms of computing power, memory/storage size, display resolution, display frame rate, display color gamut, etc. Network and transmission channels also have widely varying characteristics in terms of packet loss rate, available channel bandwidth, burst error rate, delay, etc. Moreover, much video data today is transmitted over a combination of wired and wireless networks, further complicating the underlying transmission channel's characteristics. In such scenarios, the premise of scalable video coding provides an attractive solution to improve the quality of experience for video applications running on devices with different capabilities and/or over heterogeneous networks. Scalable video coding encodes the signal once at highest representation (temporal resolution, spatial resolution, quality, etc), but enables decoding from subsets of the video streams depending on the specific rate and representation required by certain applications running on specific client devices [4]. Compared to simulcast, scalable coding can save backbone network bandwidth and storage requirements, and provide enhanced error resilience. The international video standards MPEG-2 Video, H.263, MPEG4 Visual and H.264 all have tools and/or profiles that support some modes of scalability. The open source codec VP-9, mainly developed by Google, also announced its scalable extensions recently because scalable coding fits well with real-time communication (WebRTC). Scalable Video Coding (SVC) is the extension of H.264 (Annex G of [3]). SVC enables the transmission and decoding of partial bitstreams to provide video services with lower temporal or spatial resolutions or reduced fidelity while retaining a relatively high reconstruction quality given the rate of the partial bitstreams [6]. One significant design feature of SVC is called Single Loop Decoding, which refers to

the fact that an SVC decoder only needs to set up one motion compensation/deblocking loop at the layer being decoded, and does not have to set up motion compensation/deblocking loop(s) at other lower layer(s). Thus, SVC does not require a reference picture from lower layers to be fully reconstructed, reducing computational complexity and memory requirements at the decoder. Single loop decoding is achieved by constrained inter-layer texture prediction, where, for a current block in a given layer, spatial texture prediction from a lower layer is permitted only if the corresponding low layer block is coded in intra mode (this is also called restricted intra prediction). To further improve rate-distortion efficiency of an enhancement layer (EL), SVC uses additional inter-layer prediction techniques such as motion vector prediction, residual prediction, mode prediction, etc., from lower layers.

High Efficiency Vide Coding (HEVC) [1][2], specified by the Joint Collaborative Team on Video Coding (JCT-VC) from ITU-T Video Coding Experts Group and ISO/IEC Moving Picture Experts Group, was finalized in early 2013. HEVC is expected to be deployed quickly because it can save about 50% bitrate compared to its predecessor, H.264/AVC [3], with increased bit rate savings for higher resolution video. After completing the first version of the High Efficiency Video Coding (HEVC) standard [1], ITU-T VCEG and ISO/IEC MPEG jointly issued the call for proposals for the scalable extension of the HEVC standard [7] (also abridged as SHVC). Unlike SVC, which is based on block level inter layer prediction design, SHVC is developed by changing high level syntax to achieve various video scalability requirements [5]. Such high level syntax-based design applies the inter-layer prediction process to lower layer reconstructed pictures to obtain the inter-layer reference pictures. Then, to predict higher layer pictures, the inter-layer reference pictures are used as additional reference pictures without resorting to block level syntax changes. Compared to prior scalable video coding standards, SHVC can be more easily implemented as the overhead of architecture design is largely reduced. Another feature of SHVC is hybrid scalable coding [15], which allows the base layer (BL) pictures to be coded using a legacy standard such as H.264/AVC or MPEG-2. The hybrid scalability feature can provide efficient video services to users with legacy devices and users with new devices simultaneously. Color gamut scalability is another new functionality of SHVC, which supports efficient scalable coding when different color spaces are used in the base and enhancement layers.

Compared to H.264 SVC, SHVC adopts an easily implemented multi-loop coding framework to possibly re-use the existing HEVC codec design [15]. The low level process of each enhancement layer of the SHVC codec is kept the same as a single layer HEVC codec, and only high level syntax (HLS-only) changes are applied at enhancement layer for inter-layer processing, operation point signaling, etc. The HLS-only architecture adopted by SHVC allows implementations to make maximal reuse of existing HEVC design. Inter-layer processing of the reconstructed reference layer pictures, including inter-layer texture prediction as a so-called “reference index” approach [16] and inter-layer motion prediction [17], is applied to improve the coding efficiency of enhancement layers. Figure 1 shows the SHVC encoder and decoder with two layers to support UHD and HD simultaneously. Reconstructed motion and texture from the base layer are processed by inter-layer processing for enhancement layer inter-layer prediction. The inter-layer process includes motion mapping, texture up-sampling, color gamut conversion and inverse tone mapping to compensate for the difference in spatial resolution, color space and bit-depth between the base layer and enhancement layer.



(a) SHVC encoder with two layers (b) SHVC decoder with two layers

Figure 1. SHVC encoder and decoder with two layers (HEVC* refers to HEVC with HLS changes at EL)

Real-time decoding is challenging when the picture spatial resolution and/or temporal frame rate increases, such as for HD 1080p or UHD 4K. Real-time decoding becomes more difficult for a multiple layered video codec such as SHVC. HEVC [1] has some high-level parallel processing tools to accelerate decoding speed at the cost of small coding efficiency loss, such as wavefront parallel processing (WPP), tiles and slices. WPP allows the entropy decoding of each CTU row to be processed in parallel with a small 2-CTU processing delay, which is due to CABAC context synchronization between the current CTU row and its top CTU row. Tile allows the decoder to decode each tile in parallel within one slice. Slice allows the decoder to process each slice in parallel within one picture. A real-time SHVC decoding method was proposed in [10][11] based on open source project OpenHEVC [13]. The test bitstreams were generated with the WPP coding tool. The method in [10][11] uses two parallel processing methods for decoding acceleration: picture level parallel decoding and CTU row based parallel decoding with WPP. The picture level parallel processing has two drawbacks: 1) it cannot be applied for delay sensitive applications, due to the additional one group of picture (GOP) latency that it causes; and 2) it requires more memory in order to store more decoded pictures in the decoded picture buffers. This WPP based CTU row parallel processing method can be only applied to those bitstreams with WPP enabled.

In this paper, we present a real-time parallel SHVC decoder based on SHM-2.0 software [8]. All test bitstreams are generated as one slice per picture coded under common test condition (CTC) [9], and coding tools such as wavefront or tile designed for fast decoding are not enabled. In contrast to parallel decoding design at picture level or GOP level, which requires large memory size and complex design, low level parallel processing design is proposed in this paper. The proposed low level parallel processing design can be combined with other high level parallel processing technologies to achieve increased speeds.

The rest of this paper is organized as follows. In Section 2, SHVC parallel decoding framework at layer and slice level is introduced. Section 3 reports simulation results and analyzes decoder performance. Finally, conclusions are drawn in Section 4.

2. SHVC PARALLEL DECODING FRAMEWORK

Figure 2 shows the decoding time, in percentages, of BL decoding, EL decoding, and up-sampling for 1080p spatial 2x and 1.5x bitstreams for random access (RA) configuration, using SHM-2.0 decoder. From the profiling data, the following observations are made:

1) Base layer decoding takes around 20-25% of total decoding time
2) Enhancement layer decoding takes around 51-55% of the total decoding time
3) Base layer decoded picture up-sampling and motion vector up-sampling takes around 24-25% of the total decoding time.

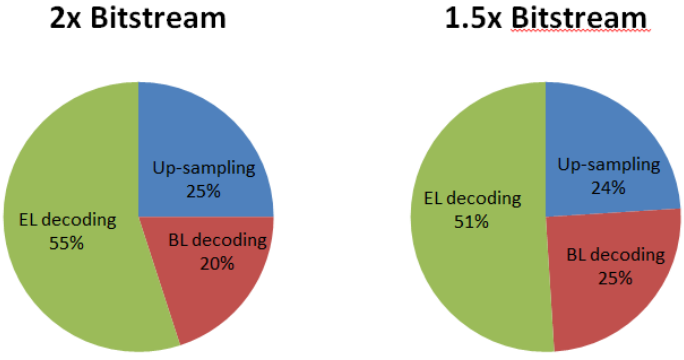


Figure 2. Processing time at various stages of SHVC decoding with SHM-2.0

The proposed implementation scheme uses multi-threading (MT) at different levels (layer, slice, and CTU-group) and SIMD optimizations to achieve real-time decoding. Two different threads are used for parallel processing of base and enhancement layer decoding at the top level SHVC decoder. The first thread of the top level SHVC decoder is responsible for decoding the base layer video, generating the inter-layer picture by up-sampling the reconstructed base layer picture and generating its motion field by up-scaling base layer compressed motion, according to the spatial ratio

information between two layers. The second thread of the top level SHVC decoder is responsible for decoding the enhancement layer video using the inter-layer picture generated by the first thread. Slice level decoding of each picture is performed in multi-threaded pipeline design at various stages like entropy decoding, motion compensation, inverse transform, and loop filtering. The inter-layer up-sampling uses multiple threads for fast processing. All those working threads are managed by a uniform thread manager.

In this paper, the proposed SHVC parallel decoding framework achieved real time decoding using the following technologies:

1) Multi-threaded processing of base layer decoding, enhancement layer decoding and up-sampling of base layer reconstructed picture;
2) Multi-threaded decoding of each slice with pipeline design by decomposing the decoding process into two stages: entropy decoding stage and reconstruction stage;
3) Multi-threaded loop filtering by decomposing the loop filtering to horizontal filtering cascaded with vertical filtering;
4) SIMD optimizations for motion compensation, inverse transform and de-quantization

The proposed framework contains two thread managers, namely the application thread manager and the internal decoding thread manager. High level framework of the proposed parallel decoding system is depicted in Figure 3.

The application thread manager controls the base and enhancement layer decoding threads and synchronizes the two threads of decoding process such that the enhancement layer decoding of a particular frame starts after the inter-layer reference picture up-sampled from the co-located base layer reconstructed picture is ready. The decoding process using the application thread manager is synchronized as below:

- Base layer decoding module starts decoding the base layer NAL units, based on the availability of empty picture buffers in the inter-layer reference (ILR) picture buffer list;
- The application thread manager sends an event to the base layer thread to notify the availability of the empty picture buffer in the ILR picture list if a free ILR picture buffer becomes available;
- The base layer decoding thread notifies the application thread manager of the ILR picture buffer available event after decoding and up-sampling of the base layer reconstructed picture;
- The application thread manager sends an event to the enhancement layer decoding thread about the ILR reference picture buffer availability;
- The enhancement layer decoding thread starts the decoding process after getting notification of the ILR reference picture buffer availability;
- The enhancement layer decoding thread notifies the application thread manager about the EL picture decoding completion, after finishing decoding the EL picture;
- The application thread manager notifies the base layer decoding thread about the availability of empty ILR picture buffers. This notification event will resume the base layer decoding if base layer thread is waiting for the empty ILR picture availability.

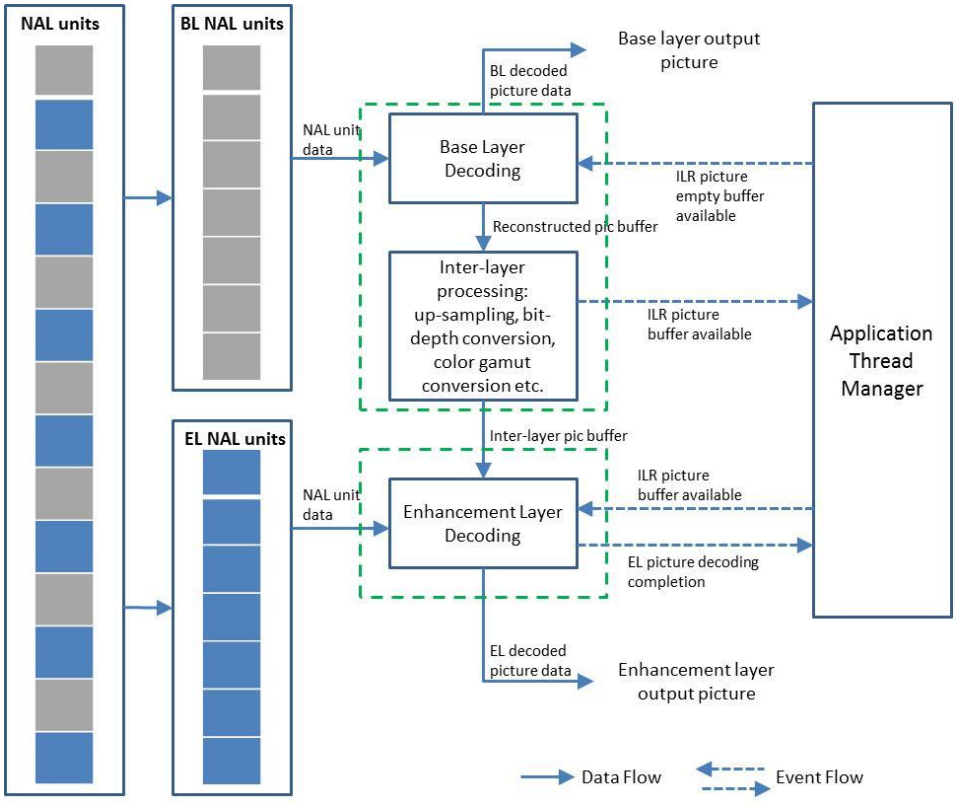


Figure 3. Parallel decoding framework

The internal decoding thread manager controls and synchronizes the slice level decoding threads like entropy decoding, motion compensation, inverse transform, and loop filtering. The layer decoding thread will create an internal decoding thread manager in both the BL and EL, namely the base layer thread manager and enhancement layer thread manager, respectively. The base layer thread manager additionally schedules up-sampling of the base-layer reconstructed picture. The internal decoding thread manager can use different numbers of threads; the number of threads used can be configured during the creation of the video decoder. The multi-threaded slice decoding process is shown in Figure 4.

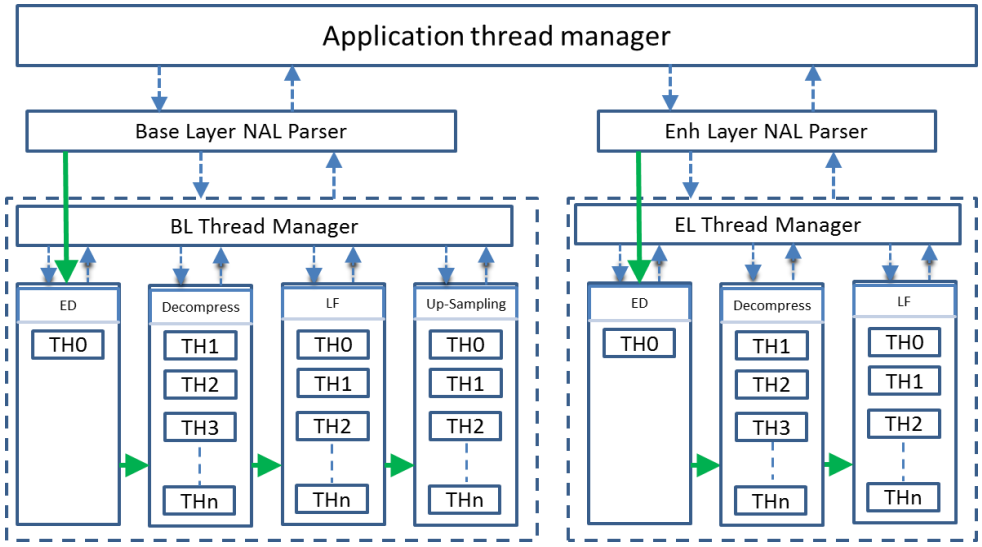


Figure 4. Multi-threaded slice decoding for base and enhancement layer

2.1 Multi-threaded processing at top level decoder

The proposed solution handles the parallel decoding process using base layer and enhancement layer decoding threads. The base layer thread is responsible for decoding the base layer data and producing the inter-layer reference picture by up-sampling the base-layer reconstructed picture and up-scaling its motion vectors with motion field mapping. The enhancement layer thread decodes the enhancement layer data referring the inter-layer reference picture and motion information provided by the base layer decoding thread. A list of inter layer reference pictures along with the motion vectors are maintained at the top level decoder, to make enhancement layer decoding in parallel with base layer decoding. This technique can be used to reduce the waiting time of the enhancement layer decoding thread, as the base layer can continue decoding and delivering the ILR pictures to the ILR picture list until the ILR picture list is full. With this technique, we were able to achieve 42-49% of overall decoding speed improvement compared to the SHM-2.0 reference decoder.

2.2 Multi-threaded processing at each slice level

The main decoding modules, such as entropy decoding, motion compensation, inverse transform, reconstruction, loop filtering and base layer reconstructed picture up-sampling, are handled in the proposed solution using multi-threaded parallel processing technique as mentioned below.

2.2.1 Parallel processing of entropy decoding and decompression

For a given layer and a given slice, the internal decoding thread manager maintains all the threads used for parallel processing. One thread will be used for entropy decoding of each layer. Other threads will be used for decompression (motion compensation, inverse transform, de-quantization and reconstruction). All the coding tree units in each slice will be separated into groups of CTUs, with each group containing 10 to 20 CTUs configured at the time of decoder creation. The size of the CTU group is related to the speed of entropy decoding and decompression. The entropy decoding thread decodes all the CTU groups and delivers them to the decompression threads. When entropy decoding of the CTU group is finished, the decompression threads will start processing that CTU group. Each decompression thread processes one CTU group at a time instead of processing one CTU at a time to reduce the synchronization overhead between decompression and entropy decoding. With this approach, the entropy decoding and decompression activities will be processed in parallel. It is recommended to use 2 or 3 decompression threads, as the complexity of motion compensation, inverse transform and reconstruction process is relatively high compared to the entropy decoding process. Simulation results are provided in Section 3 for various bitstreams with 1, 2 and 3 decompression threads. One entropy decoding and one decompression thread is used for intra slice decoding because the prediction dependency of neighboring intra blocks prevents parallel decoding of more than 1 CTU group. All intra coding units within the inter slice will not be decompressed until all inter coding units in that slice are decompressed. Figure 5 shows the dynamic status of CTU group decoding in one slice with 1 entropy decoding thread and 3 decompression threads.

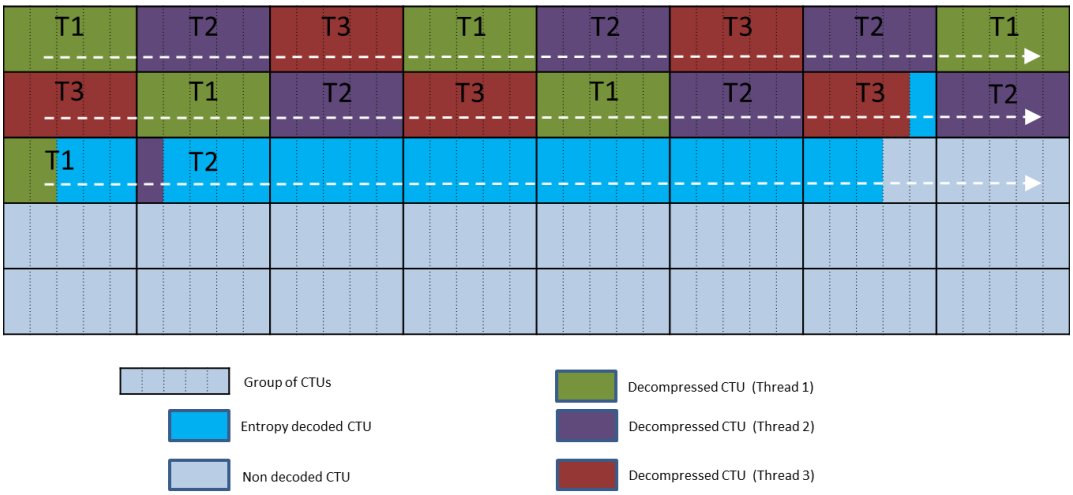
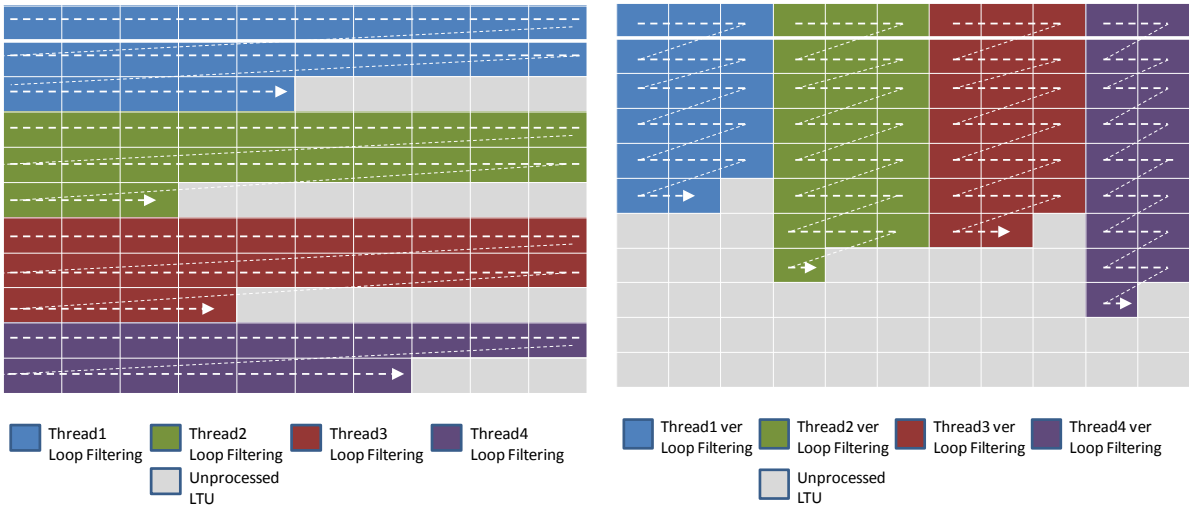


Figure 5. Multi-threaded processing of entropy decoding, decompression (MC, IT and reconstruction) within one slice

2.2.2 Parallel processing of loop filtering

After the completion of picture decompression, the thread manager uses all the threads for loop filtering. The loop filtering process is separated into two stages as horizontal and vertical loop filtering at picture level. In each filtering stage, the picture is partitioned evenly into multiple CTU regions as shown in Figure 6, and each thread will filter one region in parallel with other threads. Take Figure 6 as an example; each horizontal filtering thread will work on one set of CTU rows, and each vertical filtering thread will work on one set of CTU columns.



(a) Horizontal loop filtering (b) Vertical loop filtering

Figure 6. Multi-threaded processing of loop filtering

2.2.3 Parallel processing of inter-layer process

The up-sampling process to generate the ILR picture is divided into two sequential stages: the horizontal up-sampling stage and the vertical up-sampling stage. Multiple threads are applied to each stage. The picture is partitioned into multiple regions as shown in Figure 7. Each thread will work on one partition for horizontal or vertical up-sampling. Horizontal up-sampling is applied on the base layer reconstructed picture and vertical up-sampling is applied on the output of horizontal up-sampling.

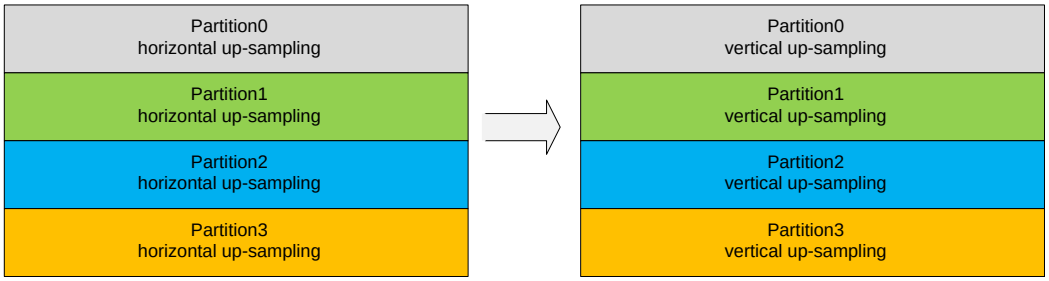


Figure 7. Multi-threaded up-sampling

Using the proposed multi-thread optimization technique at slice level, we were able to achieve 35-47% of overall decoding performance improvement with 4 concurrent threads per layer, when compared with the decoding using one thread per layer.

2.3 SIMD optimization

The most time-consuming modules like motion compensation, inverse transform, de-quantization and reconstruction are optimized using SSE4 SIMD instructions [12]. SIMD instructions use 128 bit registers instead of 32 bit registers for every operation, resulting in improved decoding speed. Memory optimization to reduce the number of memory allocations and de-allocations during slice level and CU level decoding process is applied. Using the SIMD optimization technique, we were able to achieve 39-50% of overall decoding speed improvement.

3. SIMULATIONS

Figure 8 shows the overall SHVC decoding speed achieved (in frames per second) with different combinations of the optimization techniques proposed in Section 2 for SHM-2.0 encoded bitstreams at different bitrates. The combinations of optimization techniques are as follows:

1) Step 1 optimization: Multi-threaded parallel processing technique at layer level decoding
2) Step 2 optimization: Step 1 optimization + SIMD optimization of various decoder modules
3) Step 3 optimization: Step 2 optimization + Slice level multi-threaded parallel processing.

The BL and EL parallel decoding can double the decoding speed compared to the SHM-2.0 decoder. SIMD optimization can increase speed by a factor of two. Slice level parallel processing improves decoding speed effectively, especially for medium and low bitrate range.

Figure 9 (a) and (b) are the decoding time profiling results of optimized SHVC decoder for 1080p 2x decoding. At each layer, entropy decoding for all blocks and decompression of inter-blocks takes about 60% of decoding time, and loop filtering takes about 15% of decoding time. Sample Adaptive Offset (SAO) takes about 10% because it is not optimized with multi-threads in our implementation. The decoding time percentage of intra block decompression at the base layer is much higher than that at the enhancement layer. This is because, due to the additional ILR picture available for enhancement layer coding, the percentage of intra blocks at enhancement layer is reduced compared to that at the base layer.

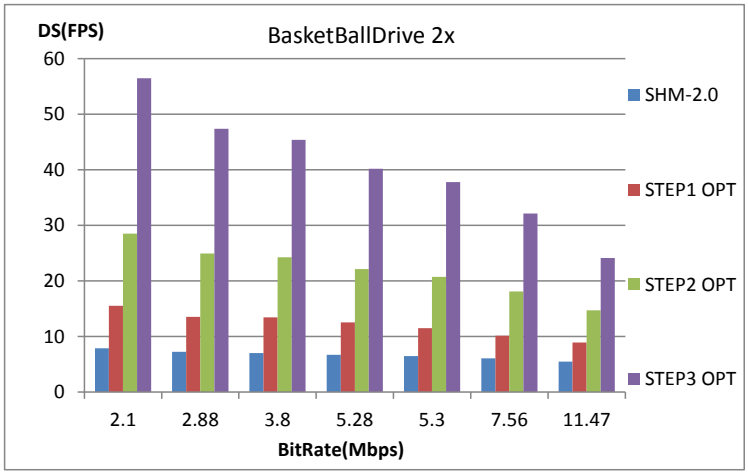
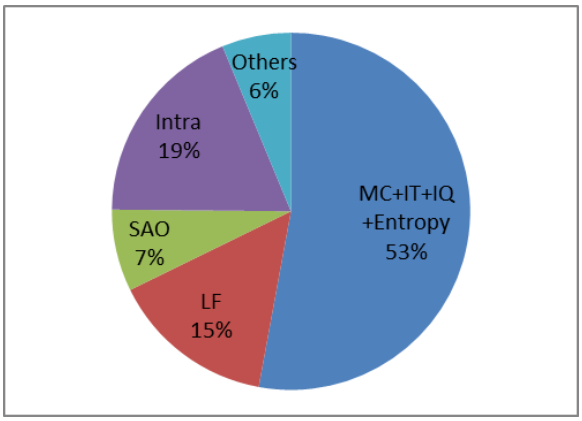
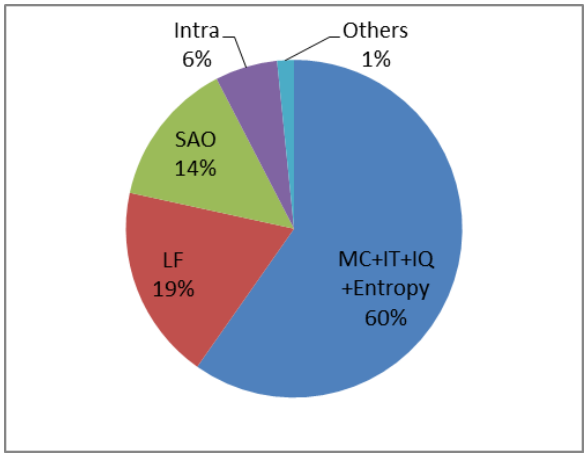


Figure 8. Decoder performance with different optimization technologies



(a) Base layer



(b) Enhancement layer

Figure 9. Profiling of optimized decoder

3.1 Decoder performance analysis

The SHM-2.0 [8] reference software is used to encode the test bitstreams using the random access configuration in SHVC CTC [9]. Three 1080p test sequences with spatial scalability 2x and 1.5x are used for analysis. The profiling tests are carried out with one entropy decoding thread and various numbers of decompression threads in parallel for both base layer and enhancement layer decoding. The simulation platform is a desktop with an Intel Core i7 2600 processor running at 3.4 GHz.

Table 1 illustrates the decoding performance of our optimized SHVC decoder for spatial 2x scalability configuration. The decoding speed with four concurrent threads exceeds 35 fps and could reach 60 fps at low bitrate. The decoding speed of the spatial 1.5x configuration is provided in Table 2. For 1.5x, the proposed solution is able to achieve decoding speeds of up to 50 fps.

Table 1. Performance of optimized SHVC decoder for 1080p 2x bitstreams

Test Sequence	BL QP	EL QP	Bite Rate (Mbps)	Decoding Speed (FPS)				
				Threads (1+3)x2	Threads (1+2)x2	Threads (1+1)x2	Thread (1)x2	SHM-2.0 reference decoder
Basketball Drive 25 fps	26	26	5.302	37.8	35.67	27.77	20.7	6.5
	24	26	5.275	40.2	36.64	29.38	22.1	6.7
	26	28	3.8	45.39	42.65	32.87	24.25	7
	30	30	2.878	47.41	44.2	33.17	24.9	7.24
	30	32	2.077	56.5	51.48	38.68	28.5	7.87
Kimono 24 fps	26	26	2.733	52.2	46.37	33.14	24.3	7.16
	30	30	1.528	60.33	53.1	37.82	27.7	7.8
Park Scene 24 fps	26	26	4.508	43.65	39.62	29.63	22.2	6.9
	26	28	3.023	50.24	45.65	33.21	24.4	7.2
	30	30	2.143	54.61	48.92	34.92	26.9	7.7
	30	32	1.561	59.76	53.77	37.57	29.6	8.1

Table 2. Performance of optimized SHVC decoder for 1080p 1.5x bitstreams

Test Sequence	BL QP	EL QP	Bite Rate (Mbps)	Decoding Speed (FPS)				
				Threads (1+3)x2	Threads (1+2)x2	Threads (1+1)x2	Thread (1)x2	SHM-2.0 reference decoder
Basketball Drive 25 fps	28	26	5.268	35.58	34.39	27.00	18.00	5.6
	28	28	3.747	41.94	40.24	31.59	19.26	6.08
	30	28	3.843	40.92	36.79	29.64	19.98	5.84
	30	30	2.763	46.47	44.11	34.45	20.70	6.34
	30	32	1.99	50.63	48.01	41.47	21.17	7.09
	32	32	2.058	49.24	47.3	36.67	21.48	6.57
Kimono 24 fps	26	26	2.75	48.48	44.86	33.71	20.65	6.23
	30	30	1.54	54.55	49.92	37.65	22.43	6.65
Park Scene 24 fps	26	24	6.25	35.8	33.37	25.27	17.83	5.60
	26	26	4.518	40.79	38.06	29.15	19.28	5.87
	28	26	4.451	39.77	37.37	28.59	19.57	5.90
	30	30	2.489	49.22	45.21063	33.95	21.43	6.45
	30	32	1.681	53.05	51.38	39.71	21.89	6.99

Figure 10 and Figure 11 illustrate the decoding performance of the proposed SHVC decoder versus the number of decoding threads for various bitstreams. As shown, decoding speed increases substantially with more threads used. The number of threads "0" refers to the SHM-2.0 reference decoder. When the bitrate is high, entropy decoding becomes the bottleneck of the whole decoding process because it is done with only one thread in our current design. The throughput of entropy decoding can be improved with advanced parallelization coding tools such as WPP or tiles.

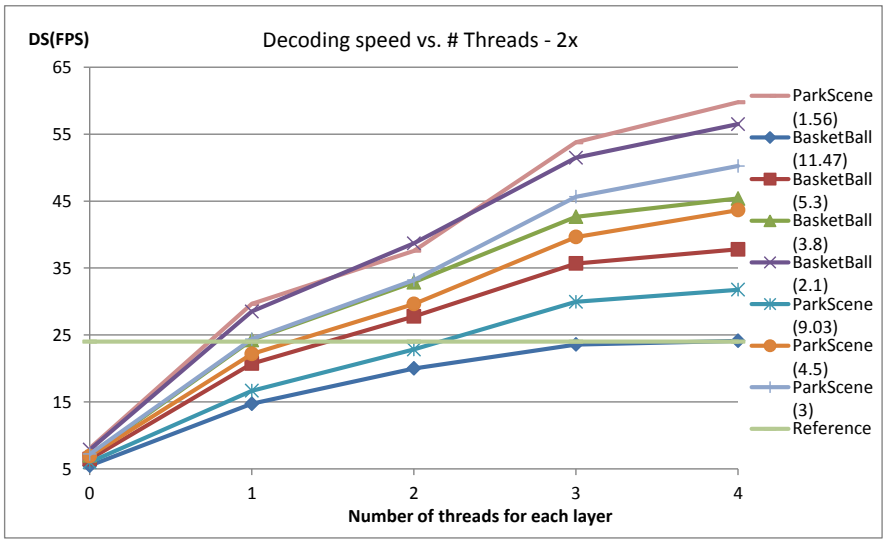


Figure 10. Decoding speed vs. number of threads used for each layer for 2x bitstream decoding at different bitrates (Mbps)

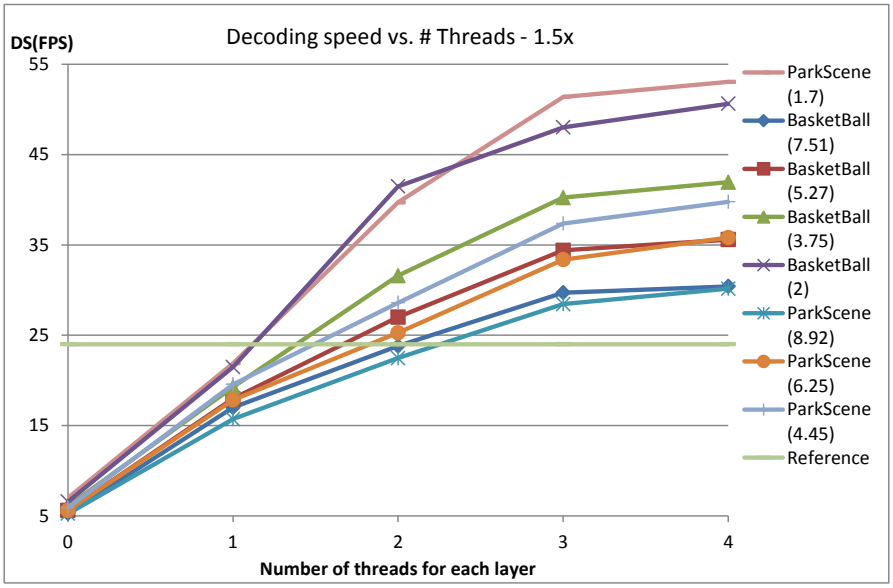


Figure 11. Decoding speed vs. number of threads used for each layer for 1.5x bitstream decoding at different bitrates (Mbps)

Figure 12 and Figure 13 show the statistics of CPU usage and memory usage, respectively, of the proposed decoder for 1080p 2x and 1.5x decoding, as a function of different numbers of threads. The number of threads "0" refers to the SHM-2.0 reference decoder. As shown, CPU usage increases with more threads. Even for the four threads per layer configuration, CPU occupation is only about 50%. The memory usage increases slightly when more threads are used. The proposed decoder uses less memory compared to SHM-2.0 because of applied memory optimization.

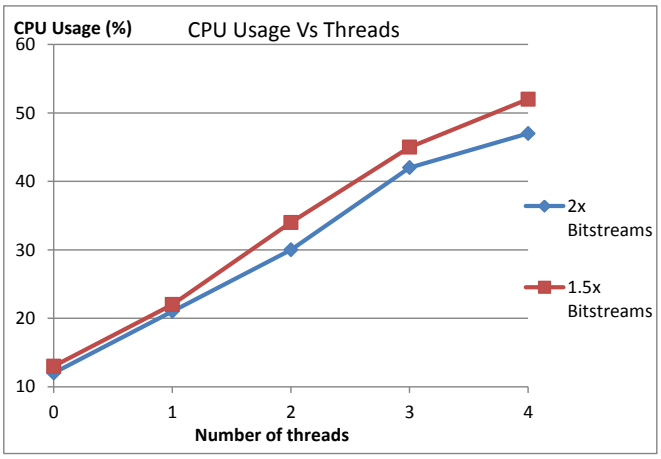


Figure 12. Average CPU usage vs. number of threads for each layer

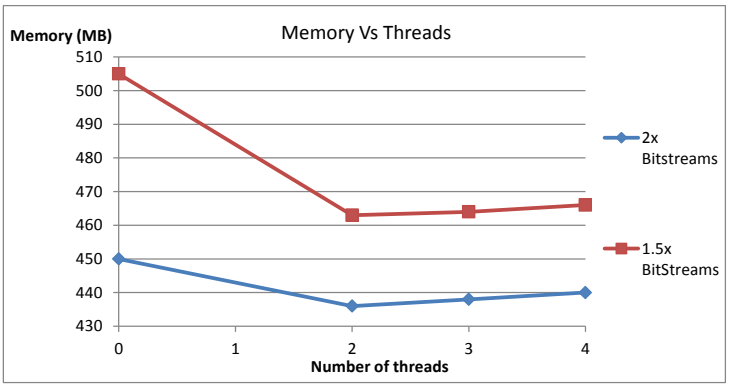


Figure 13. Peak memory usage vs. number of threads for 1080p 1.5x and 2x

The decoding speed vs. bitrate curves of two sequences (BaseketballDrive and ParkScene) with different numbers of decoding threads for 1080p 2x and 1.5x are shown in Figure 14 and Figure 15, respectively. The (1+n) in Figure 14 and Figure 15 indicates the setting of thread allocation for each layer: 1 thread for entropy decoding; n threads for decompression (MC, IT, IQ and Reconstruction); (1+n) threads for loop filtering. In base layer processing, (1+n) threads are used for up-sampling. There are primarily two reasons for why decoding speed decreases quickly when bitrate increases: 1) entropy decoding is done by a single thread in our implementation, therefore entropy decoding throughput will slow down at higher bitrates; 2) decompression also slows down at higher bitrates because there are more non-zero residual coding blocks, which require more time to decode.

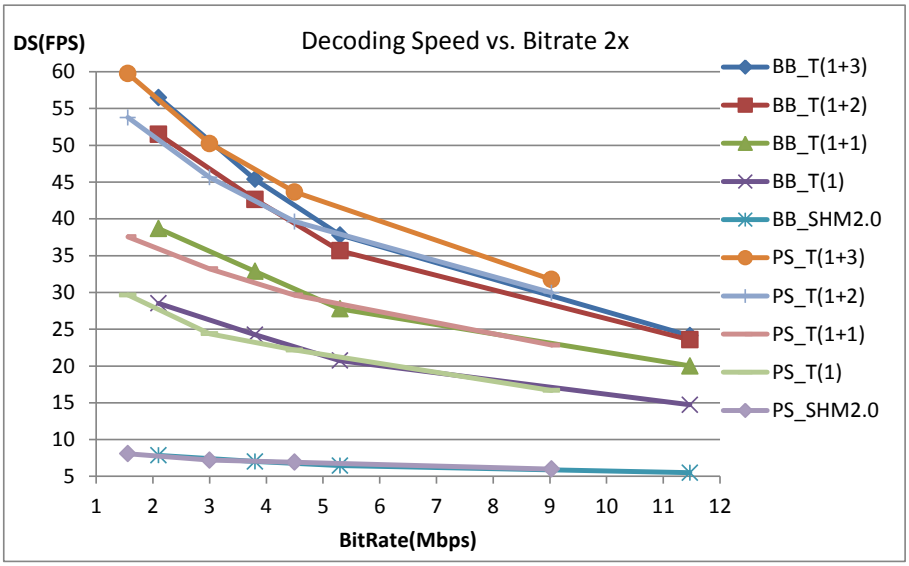


Figure 14. Decoding speed vs. bitrate for 1080p 2x bitstreams (BB: BaseketballDrive, PS: ParkScene)

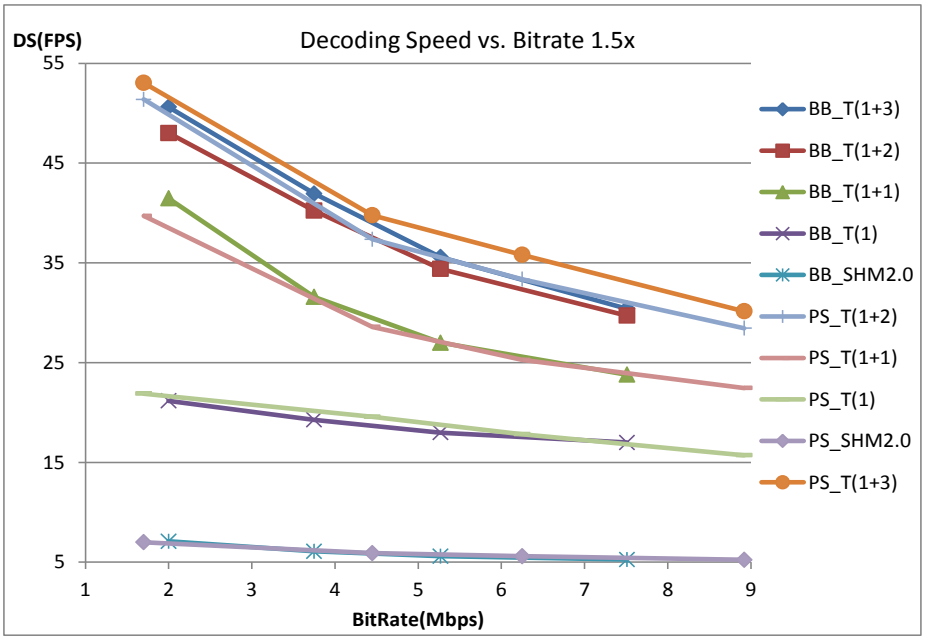


Figure 15. Decoding speed vs. bitrate for 1080p 1.5x bitstreams (BB: BaseketballDrive, PS: ParkScene)

4. CONCLUSION

This paper discusses a real-time software-based SHVC decoder using multi-threaded parallel processing. The proposed multi-threaded framework achieves decoding speeds of up to 60 fps for 1080p RA 2x and 50 fps for 1080p RA 1.5x using four threads for each layer decoding, on a desktop platform with an Intel i7 processor 2600 running at 3.4 GHz. The proposed technology can be applied to speed up SHVC bitstream decoding, regardless of whether high level parallel coding tools were enabled when these bitstreams were generated. The average CPU usage of the proposed method is about 50% at the highest decoding speed. Thus, additional multi-threading using the remaining computation resources can be applied to further improve decoding speeds for those bitstreams generated using high level parallel decoding technologies, such as WPP, tiles and slices.

REFERENCES

[1] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Trans. Circuits. Syst. Video Technol., vol.22, no.12, pp.1649-1668, Dec. 2012.

[2] B. Bross, W.-J. Han, J.-R. Ohm, G. Sullivan, Y.-K. Wang, T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Consent)," JCTVC-L1003_v34, 12th Meeting, Geneva, CH, 14–23 Jan. 2013.

[3] ITU-T Rec H.264 and ISO/IEC/MPEG 4 part 10, Advanced video coding for generic audiovisual services, November 2007.

[4] A. Luthra, J.-R. Ohm, J. Ostermann (Editors), "Use cases of the scalable enhancement of HEVC," ISO/IEC JTC 1/SC 29/WG 11 (MPEG) Doc. N12955, Stockholm, Sweden, July 2012.

[5] A. Luthra, J.-R. Ohm, J. Ostermann (Editors), "Requirements of the scalable enhancement of HEVC," ISO/IEC JTC 1/SC 29/WG 11 (MPEG) Doc. N12956, Stockholm, Sweden, July 2012.

[6] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable extension the H.264/MPEG4 AVC video coding standard," IEEE Trans. Circuits. Syst. Video Technol., vol. 17, no. 9, pp. 1103–1120, Sept. 2007.

[7] ISO/IEC JTC-1/SC29/WG11, "Joint Call for Proposals on Scalable Video Coding Extensions of High Efficiency Video Coding (HEVC)," w12957, July 2012.

[8] SHM-2.0 reference software, https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/tags/SHM-2.0.

[9] X. Li, J. Boyce, P. Onno, Y. Ye, "Common SHM test conditions and software reference configurations," JCTVC-M1009, Apr. 2013.

[10] W. Hamidouche, M. Raulet, O. Deforges, "Real time and parallel SHVC video decoder," JCTVC-N0043, 14th meeting, Vienna, AT, 25 July – 2Aug. 2013.

[11] W. Hamidouche, M. Raulet, O. Deforges, "Pipeline and parallel architecture for the SHVC decoder," 15th Meeting, Geneva, CH, 23 Oct. - 1 Nov. 2013.

[12] Intel® Architecture Instruction Set Extensions Programming Reference 319433-015, JULY 2013.

[13] OpenHEVC decoder: https://github.com/OpenHEVC/openHEVC.

[14] J. Chen, J. Boyce, Y. Ye, M. Hannuksela, G. J. Sullivan, Y.-K. Wang, "High efficiency video coding (HEVC) scalable extension Draft 6," JCTVC-Q1008, Apr. 2014, Valencia, ES.

[15] G. J. Sullivan, J. M. Boyce, Y. Chen, J.-R. Ohm, A. Segall and A. Vetro, "Standardized Extensions of High Efficieny Video Coding (HEVC)," IEEE Journal of Selected Topics in Signal Processing, Vol. 7, No. 6, Dec. 2013.

[16] J. Dong, Y. He, Y. He, G. McClellan, E.-S. Ryu, X. Xiu and Y. Ye, "Description of scalable video coding technology proposed by InterDigital," Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-K0034, 11th Meeting, Shanghai, China, Oct. 10-19, 2012.

[17] X. Xiu, Y. He, Y. He and Y. Ye, "TE C5: Motion field mapping," Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-L0052, 12th Meeting, Geneva, Switzerland, Jan. 14-23, 2013.