REAL-TIME AND PARALLEL SHVC HYBRID CODEC AVC TO HEVC DECODER

Pierre-Loup Cabarat *Wassim Hamidouche* *Olivier Déforges*

IETR / INSA Rennes (France)
pcabarat, whamidouche & odeforges@insa-rennes.fr

ABSTRACT

Scalable High efficiency Video Coding (SHVC) is the scalable extension of the latest video coding standard High Efficiency Video Coding (HEVC). One of the key novelties introduced by SHVC is that it enables hybrid codec scalability. This basically means that the video layers can be encoded with different video standards providing backward compatibility between codecs.
In this paper, we propose a software parallel SHVC decoder in hybrid codec scalability configuration. The proposed design consists of an Advanced Video Coding (AVC) decoder for the Base Layer (BL) and a HEVC decoder for the Enhanced Layer (EL). In order to perform Inter Layer Prediction (ILP), a communication of decoding states and outputs is established between the two decoders. While the native frame based parallelism is still allowed within the two decoders, the proposed design also enables the use of frame based parallelism between the two decoders. The proposed software design enables a real time decoding of the HEVC EL at 2160p60 while the AVC base layer is decoded at 1080p60 for x2 spatial scalability.
Index Terms— Real-time, SHVC, hybrid codec, HEVC, AVC

1. INTRODUCTION

Three years after its creation in 2010, the Joint Collaborative Team on Video Coding (JCT-VC) released HEVC [1] standard also known as ITU-T H.265 [2] in January 2013. HEVC is intended to take over the dominating position of H.263/MPEG-2 [3] and AVC [4, 5] for Digital Video Broadcasting (DVB) systems. However, as pointed out in [6], switching between technologies in broadcast involves lots of resources and efforts. Content providers have to think about a way to provide users with both legacy and upcoming video formats. An easy way to solve this issue appears to be simulcast: a broadcast configuration where the content is simultaneously broadcasted multiple times according to the different target formats. However, this solution of addressing the issue is not the most bandwidth efficient since the encoder does not take advantage of obvious redundancies between the multiple instances of a same content broadcasted into multiple formats (codecs). A more efficient way of dealing with this issue consists in using a multi-layers scheme, relying on scalable video coding. Multiple ELs can be used to complement a BL, each one carrying supplementary information required by the format it targets.

In this paper, we focus on SHVC hybrid codec scalability feature in the scheme of a transition from AVC to HEVC technologies. As shown in Figure 1 the two video formats can be either embedded and broadcasted in a same stream using MPEG-Transport Stream (MPEG-TS) [7] or dynamically streamed over the internet according to the available bandwidth using Dynamic Adaptive Streaming over HTTP (DASH) [8]. At the receiver's side, the HEVC EL decoding stage can then be skipped if not supported or for energy sparing purposes. At the same time, more advanced devices can benefit from enhanced quality brought by the EL supplementary data.

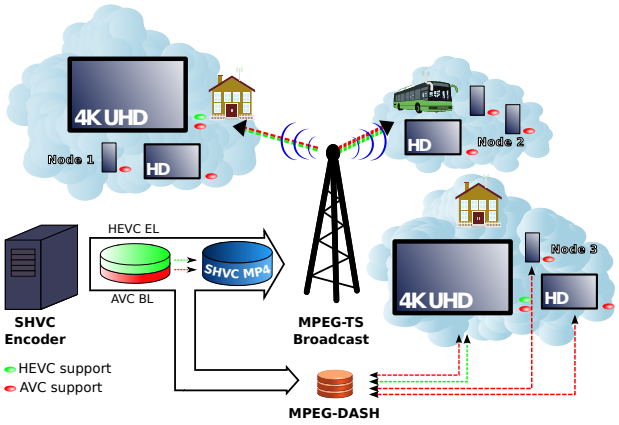

Fig. 1. Illustration of AVC/HEVC hybrid codec broadcast using SHVC

This paper provides a description of a real-time parallel hybrid codec decoder based on SHVC. To the best of our knowledge, the proposed decoder is the first realisation of a real-time AVC and HEVC hybrid codec decoder. The proposed decoder consists of a software solution based on the SHVC decoder implemented in the open source project *OpenHEVC* [9]. FFmpeg h264 [10] decoder is included into *OpenHEVC* decoder as a BL decoder. The proposed decoder benefits from the use of the multi-layers SHVC decoder design described in [11] and [12]. The decoder is optimized for different platforms and is friendly parallel to leverage multi-core processors. Thus, the decoder's design enables a real-time decoding of the multi-layer content in Ultra High Definition (UHD) at 60 frames per second.
The rest of this paper is organized as follow. Section 2 introduces the state of the art of the existing HEVC and SHVC decoders. In Section 3, we present the proposed parallel hybrid decoder design. Section 4 provides performance results of the hybrid codec decoder. Finally, Section 5 concludes this paper.

2. RELATED WORK

Finalized in July 2014, SHVC [6] is the multi-layer extension known as Annex H of the HEVC standard [1]. The SHVC extension is based on the HEVC coding and enhances the coding gain by leveraging spatial correlation between layers. As displayed in Figure 2, the SHVC decoder consists of multiple instances of the HEVC decoder

This work was supported by the 4EVER2 project (*www.4ever-2.com*)

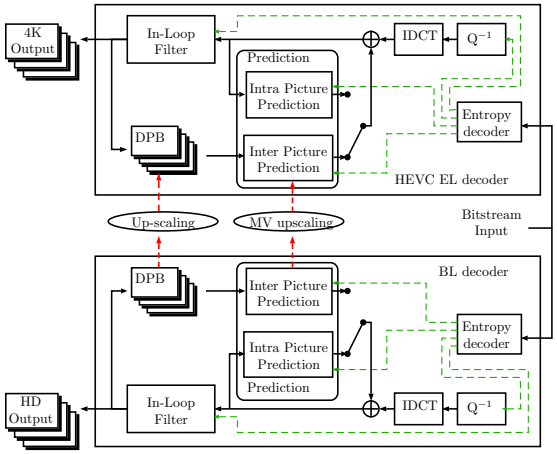

Fig. 2. Illustration of SHVC decoding process

as EL decoder, while the BL decoder can be either HEVC or raw video. The only additional operation relative to SHVC lies into the up-scaling process required by Inter Layer Prediction (ILP) in the case of spatial scalability. The up-scaling is performed directly on the raw output pictures of the direct lower layer decoder, but also on Motion Vector (MV)s when the BL corresponds to HEVC content. The up-scaled pictures and MVs can then be used as reference for Inter Picture Prediction (IPP). Therefore, the only few changes brought to HEVC standard concern High Level of Syntax (HLS) elements. The lower level of information relative to SHVC decoding can be found into to the slice headers. The decoding of Coding Tree Unit (CTU) data is still fully equivalent to the process in use for HEVC content, reducing the effort of supporting the SHVC extension from an existing HEVC implementation. In this context, the switch of technologies between HEVC and SHVC can be provided in a near-time, without intrusive modifications to be brought to existing decoder devices.

While shortening the gap between technologies, SHVC standard also introduces bit depth, color gamut and hybrid codec scalability types, which are not supported by its predecessor the Scalable Video Coding (SVC) standard [13]. The hybrid codec scalability feature proposed by SHVC proves its interest especially when considering backward compatibility issues of transiting between two codec technologies. It consists in the possibility of using different coding standards for the BL and the EL. Since the BL can be considered as raw video content, the BL can come with any coding process as soon as the correct decoder to process the BL data is available. Thus, considering the minor changes to be brought to HEVC decoders, SHVC can provide a near-time and bandwidth efficient solution to the transit between AVC and HEVC technologies.

There exist several open source software HEVC decoders such as [14–17] and [9]. Among these decoders, two support the SHVC extension namely the Scalable HEVC reference software Model (SHM) [18] and the *OpenHEVC* decoder [9]. However, the reference software decoder is not designed for real time decoding and the *OpenHEVC* does not support hybrid codec scalability configuration of SHVC. Fortunately, since SHVC decoding only requires HLS changes, those software and even existing hardware HEVC decoders such as the chip proposed in [19], or the Field Programmable Gate Array (FPGA) proposed in [20, 21], can easily be extended to support SHVC processing. As an exemple, authors in [22] already proposed a real-time SHVC encoder enabling real time encoding of 4Kp30 video in spatial scalability with a ratio of 2.

3. HYBRID CODEC DECODER DESIGN

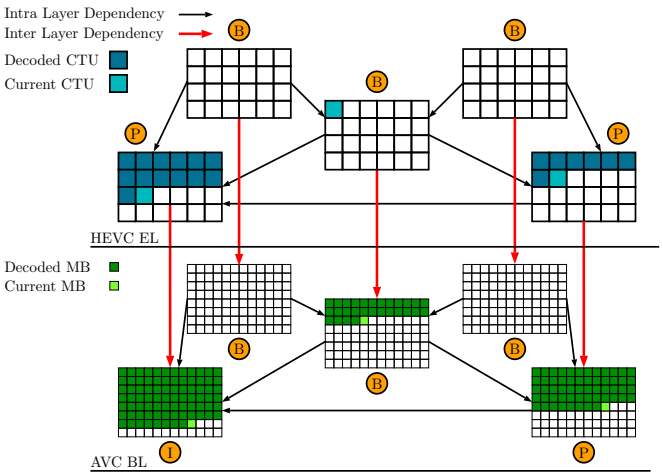

Fig. 3. Illustration of ILP frame based parallelism in the AVC/SHVC hybrid decoder in the use case of spatial scalability with a ratio of 2

The proposed hybrid codec software decoder includes the optimized AVC decoder implemented in FFmpeg library [10] within the *OpenHEVC* [9] decoder as an optional BL decoder. The decoder is based on the design of the decoder described in [11] and [12], except that the AVC does not support Wavefront Paralell Processing (WPP). We consider only one thread per frame for both BL and EL frames. Thus, it results in a simplified version of the same algorithm. Both AVC BL and HEVC EL decoders have been adapted to interact and support ILP frame based parallelism.

An insight of ILP frame-based parallelism in our SHVC decoder is illustrated in Figure 3. ILP parallelism follows the same principle than the Intra-layer frame-based parallelism, with an additional reference picture corresponding to the decoded frame of the BL. Each layer decoder is divided into multiple frame decoders, each one living into its own thread. Once a thread finishes decoding a block, it signals its position to the dependent frame threads. In Figure 3, the blue blocks correspond to already decoded HEVC CTUs by the EL decoder, green blocks correspond to the already decoded AVC Macro Block (MB)s, and clearer blocks represent the position of the block currently being decoded in each frame decoder's thread, i.e. the position of its frame's thread.

When decoding its EL current block, the frame decoder thread takes into account the progress of its inter layer reference decoding thread as well as its inner layer references decoding threads. Based on the position of its reference frame threads and the current MV, the frame decoder can determine whether the data required by Motion Compensated Prediction (MCP) is already available or not. Hence, the frame decoder thread will either wait for the reference's progress or continue the decoding of the current frame.

In order to ensure the data required by ILP is always available when decoding an EL frame, we also have to ensure that the BL decoder will not free its reference before all dependent frames are completely decoded. This is done by adding a reference to the BL frame that

will be freed when the corresponding EL frame is fully decoded. Besides, when a frame thread has finished decoding and is available to start decoding a new frame, the thread waits until a frame thread for each of the other layers are also available. These design choices enable to ensure the ELframe thread decoder will not start before its BL reference frame has begun being decoded, as well as memory usage stability. Indeed, since the BL decoder does not have any dependencies with its upper layers, it would continue decoding the following BL frames. If we suppose a BL and EL of different decoding complexities (which is almost always true, especially considering spatial scalability) the memory space used to store the BL frames required for the EL following frames' dependencies would keep growing as long as the EL decoder doesn't catch up with the BL decoder (which seems rather unlikely). Thus, by making the BL frame decoder thread stop and wait for the EL frame decoder thread to finish, we are assured we avoid an unexpected memory usage growth.

4. HYBRID CODEC DECODER PERFORMANCE

This section describes the performance of the hybrid codec decoder presented in previous section. We consider the Common Test Conditions (CTC) and reference software coding configurations [23].
The sequences described by CTC are divided into two classes. Class B consists of five 1920x1080 sequences with various frame rates and a duration of 10 seconds. Class A consists of two 2560x1600 sequences at 30 fps with a duration of 5 seconds. In order to give an insight of the expected performance for 2160p video contents, we added four 3840x2160 sequences with frame rates of 60 fps and a duration of 10 seconds into a new class noted Class U.
The Quantization Parameter (QP) values used for encoding are those described in CTC. For spatial scalability, we use BL QP values of 22, 26, 30, 34 with a delta of 0 and 2 for the EL. For SNR scalability, we use BL QP values of 26, 30, 34, 38 with deltas of -6 and -4 for the EL.
The AVC bitstreams were coded with JM-19.0 [24] reference software and the HEVC bitstreams corresponding to EL as well as their simulcast single layer equivalent were coded using SHM-9.0 reference software [18]. We only consider random access configuration and the minor changes we brought to CTC configuration files are stated hereafter. In order to obtain a ratio of 2 for spatial scalability, we consider 960x540 as a BL resolution for Class B sequences instead of 960x544. As the embedded AVC decoder does not support this configuration, we set the *BIdenticalList* parameter to 0.
All our results were carried-out on a 6 cores Intel Xeon W3670 processor running at 3.2 GHz, on Ubuntu 12.04 LTS operating system. The kernel version was 3.16.0-73 and the software was built with gcc version 4.8.4. All speed related results corresponds to an average obtained on ten decoding runs with a sleep of ten seconds after each complete decoding. It should be noted that we did not compare to [11, 12] mostly becauseWPP is not present in the AVC standard.

4.1. Bandwidth efficiency

Table 1 illustrates the spared bandwidth using different SHVC scalability types in comparison to the equivalent single layer. By single layer we mean the HEVC stream which would have been obtained in simulcast configuration using the same QP value and resolution that were used to encode the EL. Although those results do not relate to the decoder design, they are given as an additional information intended to demonstrate the interest of SHVC format AVC/HEVC hybrid codec for broadcast in the configuration we used. Given each tested scalability configuration, the HEVC EL resulting from SHVC coding using an AVC BL is compared against the HEVC stream obtained in single layer configuration. The results are given using the Bjontegaard Delta rate (BD-rate) [25] difference where negative value refers to gain of the SHVC. The BD-rate performance shows that SHVC configuration enables significant coding gains compared to the single layer configuration. Based on our results, the broadcaster could spare from 13.4% up to 47.0% of the used bandwidth.

	Sequences	x2		x1.5		SNR	
		ΔQP		ΔQP		ΔQP	
		0	2	0	2	-6	-4
Class B	Kimono	-20.4	-35.2	-35.9	-61.0	-18.7	-30.7
	ParkScene	-15.4	-24.1	-30.7	-51.3	-18.9	-30.7
	Cactus	-13.1	-21.8	-26.6	-47.6	-14.9	-25.4
	BasketBallDrive	-12.1	-22.5	-26.7	-48.8	-14.8	-26.0
	BQTerrace	-5.9	-9.8	-13.2	-26.4	-7.8	-17.8
	Average	-13.4	-22.7	-26.6	-47.0	-15.0	-26.1
Class A	Traffic	-14.8	-23.4	-	-	-15.6	-27.0
	PeopleOnStreet	-21.2	-36.5	-	-	-19.5	-34.9
	Average	-18.0	-30.0	-	-	-17.6	-31.0
Class U	Beauty	-11.4	-20.7	-	-	-13.0	-19.5
	Bosphorus	-18.5	-32.3	-	-	-15.0	-27.0
	HoneyBee	-22.5	-42.9	-	-	-15.6	-33.9
	PeopleOnStreet	-21.0	-35.4	-	-	-19.7	-34.8
	Average	-18.4	-32.8	-	-	-15.8	-28.8

Table 1. BD-Rate (in %) HEVC high resolution single layer vs HEVC EL based on AVC BL in all tested scalability configurations

4.2. Single thread performance

Class	Sequences	Frame Rate (FPS)			
		Simulcast	Multilayer		
			x2	x1.5	SNR
Class B	Kimono	69	56	31	39
	ParkScene	63	54	30	37
	Cactus	75	64	28	43
	BasketBallDrive	62	52	23	35
	BQTerrace	65	58	42	36
	Average	67	57	31	38
Class A	Traffic	23	19	-	14
	PeopleOnStreet	38	32	-	22
	Average	31	26	-	18
Class U	Beauty	19	17	-	11
	HoneyBee	26	24	-	15
	Bosphorus	20	17	-	12
	PeopleOnStreet	12	10	-	7
	Average	19	17	-	11

Table 2. Comparison of single layer per sequence single thread average frame rates (in fps) against different scalability types

Table 2 gives an insight of the decoding frame rate performance of both multi-layers and single layer schemes. It compares, in single thread configuration, the average decoding frame rates on all tested QP values for each sequence of a single layer against different multi-layer configurations. The frame rates differ within the different classes and multi-layers configurations due to the decoding of BLs of different complexity as well as the up-scaling operation which is not required by SNR scalability. However, the ratio performed

between the single layer frame rate and the multi-layers frame rates does not vary much within each scalability configuration. The speed loss involved by the decoding of the BL in each scalability type can be approximated to this ratio. The spatial scalability with a ratio of 2 introduces around 12% decoding speed loss compared to single layer decoding, while a ratio of 1.5 causes about 27% speed loss for Class B, and SNR scalability causes about 40% speed loss.

Table 3 displays the single thread frame rates decoder performance

QP_{EL}	x2	x1.5	SNR	Simulcast
20	-	-	21 fps	31 fps
22	34 fps	15 fps	27 fps	40 fps
24	41 fps	18 fps	32 fps	50 fps
26	46 fps	20 fps	35 fps	58 fps
28	52 fps	23 fps	38 fps	65 fps
30	55 fps	24 fps	41 fps	71 fps
32	60 fps	26 fps	44 fps	76 fps
34	62 fps	27 fps	46 fps	81 fps
36	66 fps	28 fps	-	85 fps

Table 3. Single thread frame rates (in fps) per EL QP values obtained on *BasketBallDrive*

obtained from *BasketBallDrive* video sequence for different QP values and different scalability configurations. It illustrates the increase in decoding speed when raising the QP value. This is most likely thanks to a lower number of residual transform coefficients when raising the QP. For QP values from 22 to 34, speed increases of nearly a half whatever it is single layer or multi-layers configuration. It is important to note that when the frame rates are presented as average frame rates, the minimum and maximum values can differ greatly given QP configurations. Another noticeable fact is the SNR scalability outperforms spatial scalability with a ratio of 1.5. Indeed, even if the BL is the largest among multi layer configuration, it does not require any up-scaling operation, which decreases the decoding complexity.

4.3. Multiple threads performance

Table 4 gives the performance of the proposed design in terms of frame rates speed up and latency. It provides average frame rates, decoding frame times, and speed-up per class and per number of threads for the different scalability types. The number of threads is denoted n, n_{BL} and n_{EL} denotes the number of threads given to the BL and the EL respectively. Note that on the six cores processor used in our experiment, results for thread numbers greater than 6 configuration relates to hyper-threading.

The results show that the decoder achieves an overall good performance in terms of frame rates, speed-up and latency. The proposed decoder is able to achieve real-time decoding in all configurations. While SNR scalability for Class U sequences seems to be the most critical, the decoder still nearly achieves 50 fps when using 12 threads, and performs up to 79 and even 155 fps on Class A and Class B sequences. This is due to the fact that although it does not require up-sampling, the BL size is the largest among all scalability types so it requires more decoding operations. In spatial scalability configuration when using a ratio of 2, decoding performance are above 60 fps for the three sequences' classes in test. Class B sequences goes up to 194 fps and Class A nearly achieve 100 fps. The speed up stays relatively close within each scalability types regarding the class of the sequence. The best speed-up and latency results are observed in SNR configuration. Since the BL and EL share the same pixel resolution, it is reasonable to think that the BL and EL decoding complexity are close. Thus the ILP frame based parallelism is the closest to inner layer frame based parallelism. This way, giving the BL and EL the same number of threads seems a rather good strategy. For spatial scalability types, the speed-up results are lower than in SNR scalability. This is also found in decoding frame times which overtake greatly those of SNR type. This relates to up-scaling operations which adds complexity to the threading communication process when computing the equivalent upscaled position of it's references. Moreover, when a BL frame thread has completed the decoding its frame and is available to decode a new frame, it has to wait an EL frame thread to be available before beginning decoding the next frame. Since in spatial scalability the BL is more likely less complex to decode than the EL, it does not benefit fully from the speed-up brought by its inner layer frame base parallelism. Then the speed up could be enhanced by using WPP configuration and enabling the EL to own more threads than the BL, aiming to bring the EL decoding frame times closer to those of the BL.

			Threading configuration $n(n_{BL},n_EL)$					
			1(1)	4(2,2)	6(3,3)	8(4,4)	10(5,5)	12(6,6)
Class A	x2	DFR	26	51	72	82	95	97
		S-U	1	2.02	2.86	3.30	3.85	3.92
		DFT	36.23	86.87	137.07	189.14	234.56	275.45
	SNR	DFR	18	40	58	65	76	79
		S-U	1	2.22	3.22	3.67	4.30	4.45
		DFT	19.08	54.26	78.86	113.90	133.23	156.64
Class B	x2	DFR	57	110	148	179	187	194
		S-U	1	1.94	2.60	3.12	3.29	3.41
		DFT	16.14	39.24	60.75	82.93	103.80	121.11
	x1.5	DFR	31	63	89	104	112	116
		S-U	1	2.03	2.88	3.37	3.65	3.78
		DFT	16.24	41.58	67.60	91.74	113.16	130.54
	SNR	DFR	38	82	120	137	149	155
		S-U	1	2.16	3.11	3.58	3.90	4.07
		DFT	9.11	25.92	37.55	49.15	58.12	64.70
Class U	x2	DFR	17	34	48	55	63	64
		S-U	1	2.01	2.85	3.26	3.83	3.89
		DFT	61.86	145.22	229.26	318.39	400.96	474.09
	SNR	DFR	11	26	36	41	46	48
		S-U	1	2.23	3.14	3.59	4.13	4.24
		DFT	29.41	86.57	132.13	187.24	223.65	266.12

Table 4. Average Decoding Frame Rate (DFR) (in fps), Speed-Up (S-U) and Decoding Frame Time (DFT) (in ms) per class, scalability configurations and number of threads

5. CONCLUSION

In this paper, we proposed a parallel hybrid codec SHVC decoder using an AVC decoder as a BL decoder and a HEVC decoder as a EL one. The decoder has been tested in various scalability and threading configurations. Our results show that the proposed decoder achieves real-time decoding up to 2160p60 on random access profile. Hence, it demonstrates that the additional complexity involved in the use of AVC to HEVC SHVC hybrid codec instead of single layer HEVC for real-time decoding, can already be overcome by existing devices. Finally, based on our results, SHVC becomes a serious candidate to the upcoming transition from AVC to HEVC into broadcasting technologies. Especially when considering the bandwidth it spares against simulcast.

6. REFERENCES

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 22, pp. 1648–1667, December 2012.

[2] ISO/IEC 23008-2 HEVC (ITU-T Rec. H.265), "High Efficiency Video Coding," January 2013.

[3] ITU-T Rec. H.263, "Video Coding for Low Bit Rate Communication," Tech. Rep., ITU-T, February 1995.

[4] ISO/IEC 14496-10 AVC (ITU-T Rec. H264), "Advanced video coding for generic audiovisual services," Tech. Rep., ITU-T, November 2007.

[5] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[6] Jill M. Boyce, Yan Ye, Jianle Chen, and Adarsh K. Ramasubramonian, "Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 20–34, Jan. 2016.

[7] ISO/IEC 13818-1 (ITU-T Rec. H262), "Generic coding of moving pictures and associated audio information," Tech. Rep., ITU-T, 1995.

[8] Stockhammer, T., "Dynamic Adaptive Streaming over HTTP Standards and Design Principles," *ACM Conference on Multimedia Systems*, 2011.

[9] "Open Source HEVC decoder openHEVC," https://github.com/OpenHEVC/openHEVC.

[10] "Open Source multimedia framework FFmpeg," https://ffmpeg.org/.

[11] W. Hamidouche, M. Raulet, and O. Deforges, "4K Real-Time and Parallel Software Video Decoder for Multilayer HEVC Extensions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 169–180, Jan 2016.

[12] W. Hamidouche, M. Raulet, and O. Deforges, "Parallel shvc decoder: Implementation and analysis," in *2014 IEEE International Conference on Multimedia and Expo (ICME)*, July 2014, pp. 1–6.

[13] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sept. 2007.

[14] K. McCann, B. Bross, W.-J. Jan, I.-K. Kim, K. Sugimoto, and G.-J. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 9 (HM 9) Encoder Description," Oct. 2012.

[15] Benjamin Bross, Mauricio Alvarez-Mesa, Valeri George, Chi Ching Chi, Tobias Mayer, Ben Juurlink, and Thomas Schierl, "Hevc real-time decoding," 2013, vol. 8856, pp. 88561R–88561R–11.

[16] "cclxv," https://bitbucket.org/prunedtree/cclxv.

[17] "libde265," https://github.com/strukturag/libde265.

[18] "SHVC Reference software model (SHM)," https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/.

[19] M. Tikekar, C. T. Huang, C. Juvekar, V. Sze, and A. P. Chandrakasan, "A 249-Mpixel/s HEVC Video-Decoder Chip for 4K Ultra-HD Applications," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 61–72, Jan 2014.

[20] D. Engelhardt, J. Moller, J. Hahlbeck, and B. Stabernack, "FPGA implementation of a full HD real-time HEVC main profile decoder," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 3, pp. 476–484, Aug 2014.

[21] M. Abeydeera, M. Karunaratne, G. Karunaratne, K. De Silva, and A. Pasqual, "4K Real-Time HEVC Decoder on an FPGA," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 236–249, Jan 2016.

[22] Ronan Parois, W. Hamidouche, M. Raulet, and O. Deforges, "Efficient Parallel Architecture of an intra only Scalable multilayer HEVC encoder," in *IEEE Conference on Design and Architectures for Signal and Image Processing 2016*, October 2016.

[23] Vadim Seregin and Yong He, "Common Conditions and Software Reference Configurations," Document JCTVC-Q1009, JCT-VC of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Valencia, ES, March 2014.

[24] "Joint Reference test Model (JM)," http://iphome.hhi.de/suehring/tml/.

[25] Gisle Bjontegaard, "Calculation of average PSNR differences between RD-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.