Parallel SHVC decoder: Implementation and analysis

Wassim Hamidouche, Mickaël Raulet, Olivier Deforges

PARALLEL SHVC DECODER: IMPLEMENTATION AND ANALYSIS

Wassim Hamidouche, Mickael Raulet and Olivier Déforges

IETR/INSA, Rennes, France
Wassim.Hamidouche@insa-rennes.fr

ABSTRACT

The new Scalable High efficiency Video Coding (SHVC) standard is based on a multi-loop coding structure which requires the total decoding of all intermediate layers. The decoding complexity becomes then a real issue, especially for a real time decoding of ultra high video resolutions.
A parallel processing architecture is proposed to reduce both the decoding time and the latency of the SHVC decoder. The proposed solution combines the high level parallel processing solutions defined in the HEVC standard with an extension of the frame-based parallelism. The latter solution enables the decoding of several spatial and temporal SHVC frames in parallel to enhance both decoding frame rate and latency. The wavefront parallel processing solution is used for more coarse level of granularity. The proposed hybrid parallel processing approach achieves a near optimal speedup and provides a good trade-off between decoding time, latency and memory usage. On a 6 cores Xeon processor, the parallel SHVC decoder performs a real time decoding of 1600p60 video resolution.

Index Terms— HEVC, SHVC, parallel processing.

1. INTRODUCTION

The Scalable High efficiency Video Coding (SHVC) standard is the scalable extension of the HEVC standard [1]. The SHVC standard is currently being developed by the ITU-T VCEG and by the ISO/IEC MPEG under a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC). The objective behind this work is to define tools to provide temporal, spatial and quality (SNR) scalability. Since the temporal scalability is already enabled in HEVC with a hierarchical temporal prediction structure, SHVC concentrates on spatial and SNR scalability. Several scalable solutions [2, 3, 4, 5] were proposed as a response to the SHVC call for proposal [6]. The approved approach is based on multi-loop decoding structure (i.e. all intermediate layers need to be decoded) and uses the same technologies of HEVC with an inter-layer prediction to improve the coding efficiency. This solution allows a gain of 20%-35% in terms of rate-distortion compared to a simulcast coding configuration. The SHVC encoder encodes the original video into L layers. The first layer represents the base quality of the video, and decoding more layers allows to further enhance the temporal, spatial or SNR quality of the video. To decode the L^{th} video layer, all intermediate layers ($l = 1, ..., (L-1)$) need to be fully decoded to perform inter-layer predictions. Moreover, in the case of spatial scalability, the decoded intermediate pictures are first up-sampled to match with the size of the upper layer picture. These extra operations considerably increase the complexity of the SHVC decoder compared to a single layer HEVC decoder. Therefore, SHVC decoding complexity becomes a real issue, especially to reach a real time decoding of ultra high video resolutions. In this paper we address the complexity-related aspects of the SHVC decoder. A hybrid parallel processing architecture is proposed to decrease the decoding time and the latency of the SHVC decoder. The proposed architecture combines the high level parallel processing solutions defined in the HEVC standard with an extension of the frame-based parallelism approach. This approach enables to decode several spatial and temporal SHVC frames in parallel to increase the decoding frame rate and reduce the frame latency. The high level parallel processing solutions, including wavefront, are used for more coarse level of granularity. The parallel SHVC decoder is based on the *OpenHEVC* software [7], which implements a conforming single layer HEVC decoder. The performance of the parallel SHVC decoder is evaluated on a computer fitter with a 6 cores Inter Xeon processor running at 3.2 Ghz. Experimental results show that the proposed solution performs a near optimal speedup and provides a good trade-off between decoding time, latency and memory usage. This paper is organized as follows. Section 2 gives a brief description of both HEVC and SHVC standards, including parallelization strategies in HEVC. Section 3 describes the parallel extension of the single layer *OpenHEVC* decoder. The SHVC decoder and the proposed hybrid parallel processing solution are presented in Section 4. The performance of the parallel SHVC decoder is assessed and discussed in Section 5

2. RELATED WORK

2.1. HEVC standard

The HEVC standard can reach the same subjective video quality as its predecessor H.264/AVC at about a half bitrate

[8]. This gain is obtained thanks to new tools adopted in the HEVC standard, such as quadtree-based block partitioning, large transform and prediction blocks, accurate intra/inter predictions and the in-loop sample adaptive offset (SAO) filter. The HEVC frame is partitioned into Coding Tree Units (CTUs). Each contains one luma Coding Tree Block (CTB) and two chroma CTBs. Recursive subdivision of a CTU results in Coding Unit (CU) leaves with the corresponding Coding Blocks (CBs). The CU can be split into Prediction Units (PUs), a basic entity for intra and inter predictions, and recursively split into Transform Units (TUs), a basic entity for residual coding [1]. The HEVC standard was designed with a particular attention to complexity, where several steps can be easily performed in parallel [9, 10]. Three high level parallel processing approaches, including independent slice, tile and wavefront, can be used in HEVC to simultaneously process multiple regions of a single picture. The frame can be partitioned into one or many slices, mainly to increase the bitstream robustness. The independent slices break the CABAC and the intra prediction dependencies and thus can be used for parallel encoding and decoding. The tile concept splits the picture into rectangular groups of CTBs, called tiles. As for slices, tiles break the coding dependencies at their boundaries, that each tile can be independently processed. However, slice and tile concepts have several disadvantages. Indeed, intra prediction limitation and resetting the CABAC probabilities decrease the coding performance in terms of rate distortion, especially for large number of tiles/slices per frame. Moreover, the in-loop filters cannot be performed in parallel at the tile/slice edges without additional control mechanism. The Wavefront Parallel Processing (WPP) solution splits the frame into CTB rows [11]. In the WPP mode, the CABAC context is initialized at the start of each CTB row. The overhead caused by this initialization is limited since the CABAC context at each CTB row is initialized by the CABAC context state at the second CTB of the previous CTB row. As illustrated in Figure 1, the decoding of each CTB row can be carried out on separate threads with a minimum delay of two CTBs between adjacent CTB rows. Therfore, the wavefront dependencies require a delay of two CTBs between adjacent CTB rows, introducing parallelization inefficiency (not all threads are used when processing the start and the end of the frame) and requiring additional communication between threads decoding adjacent CTB rows. These three high level parallel processing solutions depend on the bitstream, and can be used only when the slice, tile or wavefront tools are enabled by the encoder. The frame-level parallelism allows to simultaneously process multiples frames, whatever the coding configuration, under the restriction that the motion compensation dependencies are satisfied [12]. The frame-based parallelism also suffers from a number of limitations. The performance of the frame-based parallelism solution strongly depends on the coding structure and the ranges of the motion vectors. Moreover, the frame-based parallelism improves the decoding frame rate but not the latency and it requires extra memory usage, compared to tile and wavefront concepts.

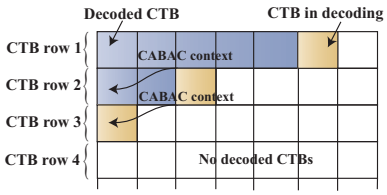

Fig. 1. Principale of the wavefront solution

2.2. Scalable SHVC standard

The SHVC standard aims to provide spatial and quality scalability with a simple and efficient coding architecture [6]. All technologies defined in the HEVC standard are used in SHVC with an inter-layer prediction to further impove the rate-distortion performance compared to a simulcast coding configuration. For spatial scalability coding configuration with L layers, the SHVC encoder consists of L HEVC encoders, one for each layer. The Base Layer (BL) HEVC encoder ($l = 1$) encodes a downsampled version of the original video and feeds the HEVC encoder corresponding to the first ($l = 2$) Enhancement Layer (EL) with the decoded picture and its motion vectors (MVs). The L^{th} HEVC encoder encodes the original video using the upsampled picture from the lower layer HEVC encoder ($l = L-1$) and its upscaled MVs as an additional reference picture for inter-layer predictions. The up-sampling operation is performed by a 8-tap interpolation filter for luma samples and a 4-tap filter for chroma samples [13]. The output of the L encoders are multiplexed to form a conforming SHVC bitstream. Therefore, the BL bitstream is HEVC conforming and can be decoded with any HEVC decoder.
In this paper, the single layer *OpenHEVC* decoder is extended to support the wavefront parallel processing solution. The analytical speedup of the WPP solution is expressed following the number of decoding threads and the video parameters. Moreover, the proposed hybrid parallel processing solution is implemented under a software SHVC decoder. The performance of the parallel SHVC decoder is assessed in different decoding configurations. To the best of our knowledge, it is the first implementation of a real time SHVC decoder enabling a hybrid parallel processing solution.

3. PARALLEL SINGLE LAYER HEVC DECODER

3.1. OpenHEVC decoder

OpenHEVC is an open source implementation of the HEVC decoder. It is written in C programming language on the top of the *FFmpeg* library [14]. The source code is heavily optimized in Single Instruction Multiple Data (SIMD) methods,

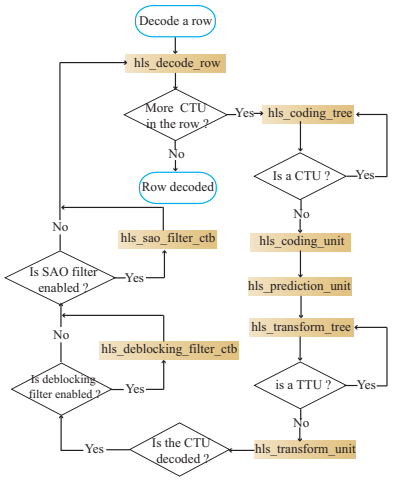

Fig. 2. Blocks diagram of the OpenHEVC decoder

including SSE Intel instructions and assembly code.
The architecture of the *OpenHEVC* decoder is based on a CTU. The decoder performs in a single pass all decoding steps at the CTU level. Figure 2 shows an overview of the *OpenHEVC* architecture. The *hls_decode_row* function decodes all CTUs of one row within the slice. It browses in raster scan the CTUs within the row and calls the recursive function *hls_coding_tree* to decode each CTU. This function browses in z-scan all CUs within the CTU and calls for each CU the *hls_coding_unit* function, performing one CU decoding. There are specific functions that handle the decoding of the prediction and the transform units, namely *hls_prediction_unit* and *hls_transform_unit*, respectively. Once all CUs within a CTU have been decoded, the deblocking filter (DF) and then the SAO filter are performed on the decoded CTU. However, when performing the DF of the current CTB, the right and the down CTB neighborhoods are not available (ie. not yet decoded). Therefore, the right and down edges of the current CTB are filtered when its right and down CTBs are being filtered, respectively. In this architecture, the DF and the SAO filters are delayed with one CTB and one CTB row for only the right and the down edges of a CTB, respectively. In terms of memory usage, the *OpenHEVC* decoder allocates two types of memory: local memory to hold informations used only at the level of the CTU, and global memory required to store informations at the picture and video sequence levels.

3.2. WPP extension in the OpenHEVC decoder

The WPP extension in the *OpenHEVC* architecture is straight forward. This is possible by running the *hls_decode_row* function on separate threads to decode several adjacent CTU rows in parallel. The delay in terms of CTU, noted d, required by the wavefront solution between two adjacent CTU rows is managed by an integer type array shared by all threads. The i^{th} value of the array is used to count the number of decoded CTUs within the i^{th} CTU row. Thus, the *hls_decode_row* function increments the related array value for each decoded CTU and decodes a new CTU only if the d next CTUs of the previous CTU row are decoded. The WPP extension requires an extra memory allocation. Each thread holds a copy of the local memory required to store informations of the CTU being decoded. Moreover, the memory of one CABAC context is allocated for each thread, and one extra CABAC context is required to save the context of the previous CTB row. Thus, if n threads are used for the decoding with the WPP solution, the memory of n copies of the local memory and $(n+1)$ CABAC contexts are allocated.

3.3. Analytical performance of the WPP solution

The analytical speedup of the WPP solution represents the upper bound of its experimental performance. Let us consider x the number of CTB columns, y the number of CTB rows and d the delay in terms of CTB between two adjacent CTB rows required by the wavefront solution. The effective number of threads n used in the wavefront solution is given as follows:

$$n = \min\left(nb\_cpu\_threads, \left\lfloor \frac{x}{d} \right\rfloor\right) \tag{1}$$

where $d \in \mathbb{N}^+$ and $nb\_cpu\_threads$ is the number of threads selected to decode the video sequence.
The analytical speedup γ is derived as follows:

$$\gamma = \begin{cases} \frac{xy}{\frac{xy}{n}+d(n-1)}, & \text{if } \alpha = 0 \\ \frac{xy}{x\left\lceil \frac{y}{n} \right\rceil + d(\alpha-1)}, & \text{if } \alpha \neq 0 \end{cases} \tag{2}$$

where $x, y, n \in \mathbb{N}^+$, and $\alpha = y \bmod n$.
The speedup of the WPP solution is equal to the number of CTBs of the frame (xy) divided by the number of CTBs decoded by each thread plus the additional delay required by the wavefront approach. When the number of CTB rows is multiple of the number of decoding threads, the delay of the wavefront solution at the decoding of the last CTBs is equal to $d(n-1)$. However, when the number of CTB rows is not multiple of the number of decoding threads, the delay at the end of the frame is equal to $d(\alpha-1)$. Figure 3 shows the analytical speedup of the WPP solution versus the number of threads for different video resolutions. We can notice that the WPP performance decreases for a large number of threads, where the additional delay considerably increases. Equation (2) does not consider the inactive threads waiting at the start and the end of decoding each frame. The waiting time of the inactive threads in terms of CTBs is computed as follows:

$$\sigma = \begin{cases} d(n^2-n), & \text{if } \alpha = 0 \\ d/2(n^2-n+\alpha^2-\alpha)+bx-d/2(b^2+b), & \text{if } \alpha \neq 0 \end{cases} \tag{3}$$

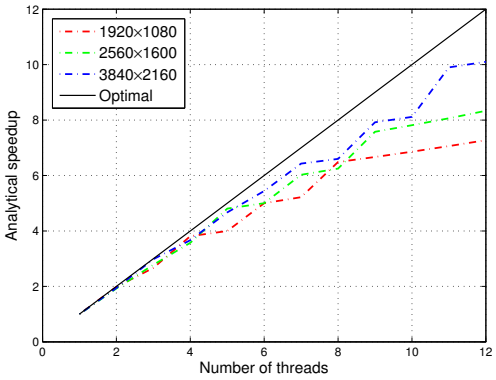

Fig. 3. Analytical speedup of the WPP solution (CTB size = 64 and $d = 2$)

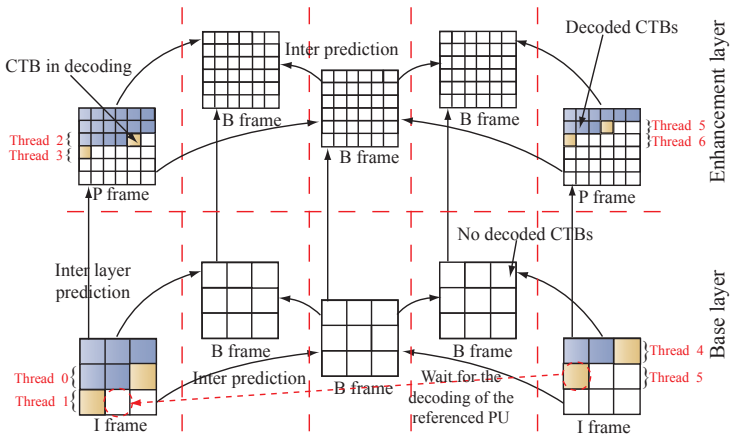

Fig. 4. Principle of the hybrid parallelism solution in the SHVC decoder: x2 spatial scalability, $L = 2$ and (n, m) = (2, 2) with n the number of threads for the wavefront solution and m the number of threads decoding different frames at each layer

where $b = n - \alpha$. We can notice from equation (3) that a high value of b considerably increases the inefficiency of the wavefront solution.

4. REAL TIME AND PARALLEL SHVC DECODER

We first implemented the SHVC decoder under the *OpenHEVC* software. The SHVC decoder consists of L instances of the *OpenHEVC* decoder, one instance for each layer, with L the number of layers. Each HEVC decoder l ($l = 1, .., L$) decodes the corresponding layer, and the l^{th} EL decoder, with $l = 2, ..., L$, has access to the picture and the MVs decoded by the lower layer decoder $(l - 1)$. Moreover, the EL decoder adds to its Decoded Picture Buffer (DPB) a new reference picture corresponding to the picture of the same temporal representation in the lower layer decoder. The inter-layer predictions are performed at the prediction unit level. Thus, the EL decoder copies the PU and the corresponding MVs from the lower layer decoder into its new reference picture. In the case of spatial scalability, the PU from lower decoder is first upsampled and its MVs upscaled before being copied into the inter-layer reference picture. Therefore, only the PUs used for inter-layer predictions are copied into the inter-layer reference picture, and upsampled in the case of spatial scalability.

The *OpenHEVC* decoder performing the decoding of each SHVC layer was extended to support the high level parallel processing solutions, including the wavefront solution. Therefore, each layer can be decoded in parallel with the wavefront approach to improve both the decoding frame rate and the frame latency. To overcome with the wavefront solution limitations and increase the parallelization efficiency, a second level of parallelism (frame-based) is introduced to the SHVC decoder. The idea behind this hybrid approach is to decode several spatial and temporal frames in parallel (frame-based solution) and decode each frame in wavefront associated with a low number of threads. This enables to take advantage of the wavefront approach at its near optimal configuration (ie. number of threads is below 4, see Figure 3). Moreover, the inactive threads waiting for both WPP and motion compensation dependencies can be used to decode other frames waiting for available threads. Figure 4 illustrates the hybrid parallelism approach in the SHVC decoder decoding two spatial scalability layers ($L = 2$). Indeed, several spatial and temporal frames are decoded in parallel under the restriction that the motion compensation dependencies are satisfied. The BL and the EL frames of the same temporal representation are simultaneously processed, which enhances both decoding frame rate and the SHVC frame latency. Since the inter-layer prediction is performed at the PU level, the BL and the EL frame can be simultaneously decoded with a control process to ensure that the PU used for inter-layer prediction is decoded at the BL decoder. The same communication process is used to decode two frames belonging to the same quality layer and different temporal representation to increase only the frame rate. This hybrid parallel processing solution combining the wavefront and the frame-based solutions in the SHVC decoder takes advantage of both solutions to improve the parallelism efficiency and reduce the decoding latency.

5. RESULTS AND ANALYSIS

5.1. Experimental configuration

The system and software configurations used to carry out the experiments are summarized in Table 1. We consider all video sequences of the SHVC common test conditions. To show the decoder performance for larger video resolution, we added to the test sequences two 3840×2160 video sequences from the

System		Software	
Processor	Intel Xeon	Compiler	GCC-4.6
	E5-1650	OS	Ubuntu 12.04
ISA	X86-64	Kernel	3.5.0-34
Clock frequency	3.2 GHz	OpenHEVC	cff4b48a94
Level 3 cache	12 MB	release	(based on HM11.0)
Cores	6		

Table 1. Configuration of the experiments

Configurations		BL + EL	BL+EL	Gain
Resolution	ratio	(SHVC)	(Simulcast)	(%)
1080p	$\times 2$	6401	1303	36
	$\times 1.5$	6432	11050	41
1600p	$\times 2$	11950	17501	31
2160p	$\times 2$	87052	135716	35

Table 2. Bit rate performance (Kbps) of the SHVC standard compared with a simulcast coding configuration (All QP)

STV High Definition Multi Format Test Set. All test video sequences were encoded with the SHVC reference software encoder [15] in two layers ($L = 2$) and two spatial scalability configurations: $\times 2$ and $\times 1.5$. The SHVC video sequences were coded in low delay coding configuration with enabling the wavefront feature where the delay between two adjacent CTB rows was set to 2 CTBs (ie. $d = 2$). The quantization parameter (QP) of the BL was set to 27 and 32, while the QP of the EL is equal to the BL QP minus 2. We consider n the number of decoding threads used for the wavefront solution, and m the number of thread used for the frame-based parallelism at each layer. To assess the performance of the hybrid parallel processing solution, we compare the performance of three decoding configurations: $(n, m) \in \{(6, 1), (1, 3), (2, 2)\}$. The SHVC decoder with the first configuration uses 6 concurrent threads for the wavefront parallelism. The second configuration decodes 3 BL and 3 EL frames in parallel. The third configuration enables the hybrid parallelism with 2 BL and 2 EL frames in parallel, and each frame uses 2 concurrent threads for the wavefront solution.

5.2. Results and discussions

Table 2 shows the bitrate performance of the SHVC encoder compared to a simulcast coding configuration for different video resolutions. The simulcast coding configuration consists in encoding the BL and the EL with two independent HEVC encoders. Therefore, the inter-layer prediction used in the SHVC standard enables in average a biterate gain up to 40% for 1080p video resolution and x1.5 spatial scalability.
Figure 5 illustrates the speedup of the wavefront implementation in the proposed SHVC decoder. The wavefront implementation achieves a speedup near to the upper bound of the wavefront solution computed in equation (2), especially when the number of decoding threads is bellow 5. With using 6 threads, the performance of the proposed implementation slightly decreases to reach a speedup of 4.5 for 2160p video resolution, instead of the upper bound value of 5.5. This is because we use the maximum number of CPU cores.
Table 3 illustrates the performance of the three considered decoding configurations in terms of speedup, decoding frame rate, decoding time per frame and memory usage. The hybrid parallel processing solution achieves the highest speedup of 4.8 and 5 for $\times 2$ and $\times 1.5$ scalable configurations, respectively. The speedup of the wavefront parallelism ($(n, m) = (6, 1)$) is decreased by the wavefront limitations, especially for small video resolutions. The high speedup performance of the hybrid parallel processing solution considerably increases the decoding frame rate, which is in average around 70 frames per second (fps), instead of 48 fps and 40 fps for the wavefront and the frame-based parallelism configurations, respectively. The decoding frame rate performance of a simulcast configuration (only one HEVC decoder) is highlighted in bold font. The single layer HEVC decoder is almost two times faster then the SHVC decoder. In simulcast configuration, only the EL is decoded without the extra complexity of the SHVC decoder related to the upsampling and upscaling processes.
The decoder latency is assessed by the time required to decode one SHVC video frame. The wavefront solution performs the lowest latency with a decoding time per frame around 30 ms. In the wavefront mode, 6 concurrent threads are used to decode the CTU rows of the frame which enables a coarse level of granularity. The hybrid decoding configuration performs lower latency then the frame based configuration thanks to the two threads used for the wavefront decoding. In terms of memory usage, the wavefront configuration requires the minimum memory since only a copy of the local memory is allocated by thread. However, the frame-based parallelism allocates a new decoder for each thread, except the DPB memory which is shared between all threads decoding the same quality layer.

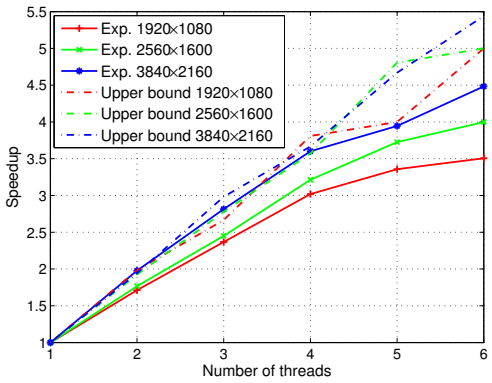

Fig. 5. Speedup performance of the WPP solution in the SHVC decoder (BL QP =27)

Configurations		Decoding configurations		
		(6, 1)	(1, 3)	(2, 2)
Speedup	$\times 2$	3.33	2.95	4.8
	$\times 1.5$	3.6	2.75	5
Decoding frame rate (fps)	$\times 2$	48	44	70
	$\times 1.5$	48	40	69
	HEVC	**99**	**74**	**116**
Decoding time per frame (ms)	$\times 2$	32	187	90
	$\times 1.5$	29	169	78
Memory usage (Ko)	$\times 2$	16881	44193	30322
	$\times 1.5$	20340	54570	37240

Table 3. Performance of the SHVC decoder in different decoding configurations (All test video sequences and the DPB memory is not considered)

6. CONCLUSION

We investigated in this paper a hybrid parallel processing solution for a real time SHVC decoder. The hybrid parallelism combines the WPP solution with an extension of the frame-based parallelism. Experimental results showed that the proposed solution performs near optimal speedup with a good trade-off between decoding time, latency and memory usage. The SHVC decoder achieves on a 6 cores Xeon processor running at 3.2 Ghz the decoding of 1600p video resolution at 60 fps. The first end-to-end video demonstration using the proposed real time SHVC decoder within the GPAC player was presented in the 106^{th} MPEG meeting in Geneva [16].

7. REFERENCES

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

[2] Z. Shi, X. Sun, and F. Wu, "Spatially Scalable Video Coding for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1813–1826, December 2012.

[3] P. Helle, H. Lakshman, M. Siekmann, J. Stegemann, T. Hinz, H. Schwarz, D. Marpe, and T. Wiegand, "A Scalable Video Coding Extension of HEVC," in *IEEE Conference on Data Compression*, March 2013, pp. 201–210.

[4] J. Chen, K. Rapaka, X. Li, V. Seregin, L. Guo, M. Karczewicz, G. V. Auwera, J. Sole, X. Wang, C. Tu, Y. Chen, and R. Joshi, "Scalable Video Coding Extension for HEVC," in *IEEE Conference on Data Compression*, March 2013, pp. 191–200.

[5] Z. Zhao, J. Si, J. Ostermann, and W. Li, "Inter-layer Intra Mode Coding for the Scalable Extension of HEVC," in *IEEE International Symposium on Circuits and Systems*, May 2013, pp. 1636–1639.

[6] ISO/IEC-JTC1/SC29/WG11 and ITU-T-SG16, "Joint Call for Proposals on Scalable Video Coding Extensions of High Efficiency Video Coding (HEVC)," in *ISO/IEC JTC 1/SC 29/WG11 (MPEG) Doc. N12957 or ITU-T SG 16 Doc. VCEG-AS90*. Stockholm, Sweden, July 2012.

[7] "Open source HEVC decoder (OpenHEVC)," in *https:://github.com/OpenHEVC*.

[8] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparaison of the Coding Efficiency of Video Coding standards including High Efficiency Video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1969–1684, December 2012.

[9] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schier, "Parallel Scalability and Efficiency of HEVC Parallelization Approaches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1827–1838, December 2012.

[10] J. F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1685–1696, December 2012.

[11] G. Clare, F. Henry, and S. Pateux, "Wavefront parellel processing for HEVC Encoding and Decoding," in *document JCTVC-F274*. Torino, Italy, Jully 2011.

[12] C. Meenderinck, A. Azevedo, M. Alvarez, B. Juurlink, and A. Ramirez, "Parallel scalability of video decoders," *Journal of Signal Processing Systems*, vol. 57, pp. 173–194, November 2009.

[13] J. Chen, J. Boyce, Y. Ye, and M. M. Hannuksela, "Scalable High Efficiency Video Coding test model 3 (SHM 3)," in *document JCTVC-N1007*. Vienna, Austria, Jully 2013.

[14] "FFmpeg: Open source and cross-platform multimedia library," in *http://www.ffmpeg.org*.

[15] "SHVC Reference Software (SHM): https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/," .

[16] W. Hamidouche, J. Le Feuvre, and M. Raulet, "A scalable HEVC demonstration within GPAC player," in *document MPEG-m31397*. Geneva, Switzerland, October 2013.