Improvement on Intra Block Copy in Video Coding with Reference Block Recompression

Lai Zhang[1], Jun Wang[2,3]✉, Jiyuan Hu[1], Pengjian Yang[1], Fan Liang[1,4]✉, Feng Lai[5]
[1]*School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China*
[2]*School of Microelectronics Science and Technology, Sun Yat-sen University, Zhuhai, China*
[3]*Southern Marine Science and Engineering Guangdong Laboratory (Zhuhai), Zhuhai, China*
[4]*Peng Cheng Laboratory, Shenzhen, China*
[5]*Zhuhai Unitech Power Technology Co. Ltd, Zhuhai, China*
{zhanglai3, hujy23, yangpj5}@mail2.sysu.edu.cn, {wangj387, isslf}@mail.sysu.edu.cn, laifeng@ut.cn

***Abstract*—Intra Block Copy is a tool adopted in HEVC and VVC extensions on SCC, it significantly improves the coding efficiency of screen content materials. To reduce memory consumption and decoder complexity, the reference region of IBC in VVC is limited. In this paper, we propose a method that recompresses the reference block of IBC, so that more reference blocks can be stored while using the same memory space. This allows IBC to expand its search range and results in BD-rate reduction. The proposed method includes spatial prediction, variable length coding, quantization, and two methods to realize the IBC buffer with recompressed reference block. We test these two methods in the reference software of VVC and achieve BD-rate reduction of 1.8% and 4.0% on average respectively for TGM test sequences.**

Keywords—Reference Block Recompression, IBC, VVC, HEVC, Lossy compression

I. INTRODUCTION

The demand for screen content video coding (SCC), such as online meetings, online education, and cloud gaming, is increasing during these years. To address the demand, many tools like Intra Block Copy (IBC), Palette Mode (PLT), Transform Skip Mode (TSM) have been adopted into SCC. Intra block copy is the one with the highest efficiency among these tools [1].

IBC is a tool firstly adopted in HEVC extensions on SCC [2] and now widely applied to most recently developed video coding standards [1] such as VVC. In IBC, block matching is performed at the encoder to find the optimal block vector (or motion vector) for each coding unit (CU). A block vector is used to indicate the displacement from the current block to a reference block, which is already reconstructed inside the current picture [3].

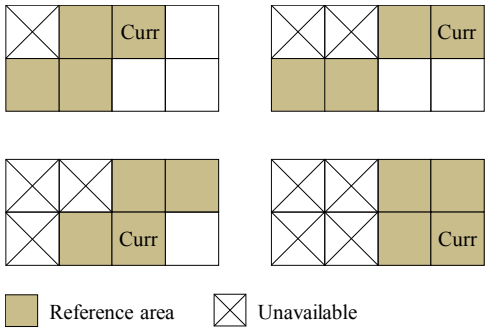

Fig. 1. Current CTU processing order and its available reference samples in current and left CTU

This work is supported by Key-Area Research and Development Program of Guangdong Province(No. 2019B010135002), and partially supported by Core & Key Industrial Technology Program of Zhuhai(No. ZH22044702190085HJL)

To reduce memory consumption and decoder complexity, the IBC in VVC allows only the reconstructed portion of the predefined area including the region of current CTU and some region of the left CTU [4], as reference area. This limitation allows the codec to use the on-chip storage as the IBC buffer. Fig. 1. illustrates the reference region of IBC Mode when the CTU size is 128, each block represents 64*64 luma sample units (the sizes mentioned later in the article all take luma sample as the unit). A 64*64 unit like this is called Virtual Pipeline Data Unit (VPDU) [3], which is defined as non-overlapping units in a picture.

However, the memory size restriction reduces the coding efficiency of IBC by about 10% [4] in BD-rate [5] reduction. To reduce the loss caused by the limited memory size, one idea is to expand the reference area without changing the size.

Inspired by Reference Frame Recompression (RFRC), which is typically used in the motion estimation and motion compensation process of inter prediction to reduce the bandwidth and power consumption in power-aware devices, we attempt to reduce the impact of memory size restriction with recompression technology. The existing recompression methods can be divided into two categories: lossless [7-13] and lossy [14-20] recompression. To the best of our knowledge, in our previous work [6], we are the first to introduce recompression methods named Reference Block Recompression (RBRC) to the IBC intra prediction. [6] proposed a lossless recompression method, which compresses the unfiltered reconstructed samples before storing them into memory and decompresses the compressed data after fetched back. This method can reduce the bandwidth on Memory Bus.

The RBRC method shows the potential in processing reference blocks of IBC, but the downside of the lossless RBRC is that the compression rate of lossless RBRC can't be guaranteed. Therefore, the compressed data needs to be stored in fixed addresses, occupying the same memory space as original data, which means that it can't save the memory size. The lossy RBRC we proposed in this article solves the problem, quantization and compression rate judgment are introduced into the RBRC to ensure that the compression rate doesn't exceed 50%. This allows the compressed data to be stored in half of the original memory space, showing in Fig. 2. Memory saving means that a buffer built with lossy RBRC

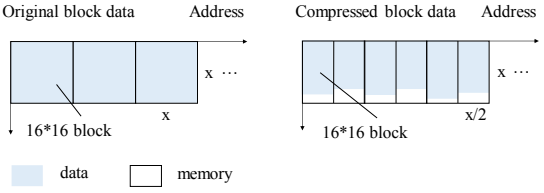

Fig. 2. The original block and the compressed block in memory

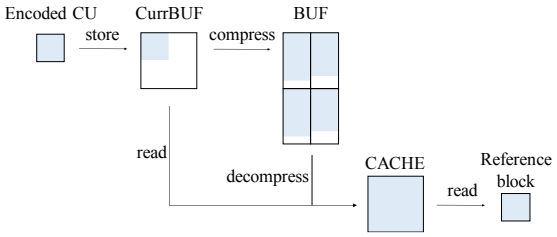

Fig. 3. The workflow of proposed IBC buffer

can store more reference blocks for the IBC under the condition of the memory size restriction, and then improves the effect of IBC and get the gain in BD-rate.

The rest of this paper is organized as follows. In Section II, the new IBC buffer with lossy RBRC is introduced. Its structure and workflow are explained. What's more, the way to achieve lossy RBRC is introduced, including spatial prediction, variable length coding, and quantization. In Section III, the experiment result of encoder and decoder with proposed method comparing to VTM 10.0 is given and explained. The conclusion is present in Section IV.

II. Proposed Method

Refer to the processing of the IBC mode in VVC, a buffer with the size of 128*128 is needed in the codec to store the reference blocks required by the IBC mode. This allows the hardware to use on-chip storage as a buffer, speeding up the IBC mode, and keep complexity low. Based on this idea, we propose to introduce Reference Block Recompression (RBRC) technology into IBC without changing the memory size set for IBC buffer in VVC, and replace the reference block in IBC buffer with the compressed reference block, showing in Fig. 2, to expand the IBC mode reference area to improve the effect of IBC mode.

A. New IBC Buffer Structure

The new IBC buffer we proposed deals with compressed data instead of the original block of samples. It is divided into three parts to store different data to help the codec to compress/decompress and read/write the reference block:

- The BUF part is used to store the blocks compressed by the RBRC method. The size is the total IBC buffer size (128*128) minus the size occupied by CurrBUF and CACHE. Its maintenance process is similar to the IBC buffer in VVC [3].
- The CurrBUF part is to store the sample of the reconstructed CU in the current VPDU during the encoding and decoding process. These samples are compressed and stored in the corresponding position in BUF after the codec enters the next VPDU.
- The CACHE part is the cache between the new IBC buffer and the prediction buffer of IBC CU. It stores the data read from CurrBUF and BUF in the past. The introduction of this part is mainly to improve the encoding speed.

These are the components of our proposed new IBC buffer with RBRC.

For the encoder, the IBC buffer needs to be accessed frequently to obtain reference blocks during the motion estimation process. To reduce the encoding time, two methods are proposed to solve this problem based on the above components, we call them method with cache and method without cache.

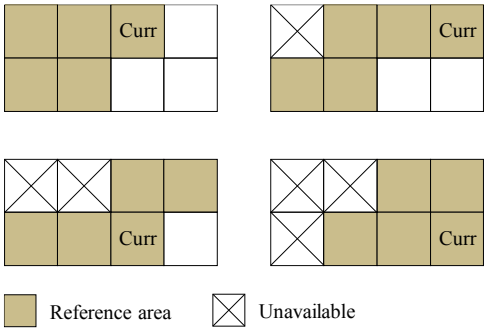

Fig. 4. Reference area of IBC with proposed method with cache

1) Method with Cache: The workflow is showing in Fig. 3. In terms of getting a reference block, the IBC CU gives the required referenced block's position and reads the data from the CACHE. If the CACHE doesn't contain the data of required block, it will decompress the required reference block from BUF or copy from CurrBUF. Then the IBC CU gets the block it needs. The CACHE reads in data that fills its own size at one time, and the required reference block is located at the upper-left of the read-in area.

In terms of storing reference blocks, the samples of reconstructed CU in the current VPDU will first be stored in the CurrBUF. When the codec is entering the next VPDU, all the samples in CurrBUF will be compressed in the unit of 16*16 compression block and stored in the BUF according to coordinate conversion rules. In the case that CU is larger than VPDU, the process of compressing VPDU will be repeated to save the reconstructed CU.

In most cases, CACHE has a square buffer area, with a side length of VPDU size, and can works for CU of any size in IBC mode. During the full search, which is used by CU not larger than 16*16, the buffer area changes into a rectangle with the high equal to 16 and the total size unchanged. This change greatly improves the utilization of cached data, since the full search is done row by row.

Under the All Intra configuration, the size of CTU is set to 128, the size of VPDU is 64*64. At this time, the size of CurrBUF and CACHE part is 64*64, and the size of BUF part is 128*64, meaning that it can store 256*64 samples. The reference area with proposed new IBC buffer is 5 VPDUs, showing in Fig. 4, and the reference area of the origin IBC mode in VVC is 4 VPDUs, showing in Fig. 1. The proposed method expands the size of the reference area by 25%.

2) Method without Cache: This method is to remove the recompression process and only retain the quantization process on the encoder side to improve the encoding speed.

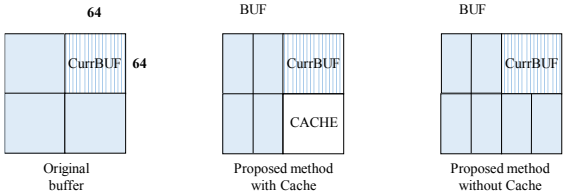

Fig. 5. Usage of 128*128 memory in different methods

The CACHE part is no longer needed. For the encoder, the BUF occupies more memory to store the quantified reference blocks without compression. which can be directly read by the encoder. For the decoder, the workflow of IBC buffer in decoder is similar to Fig. 3 but the CACHE part is discarded. All memory space other than CurrBUF is used ad the BUF part to stored the compressed reference blocks, showing in Fig. 5. This can further expand the reference area to 7 VPDUs. The quantization processes of encoder and decoder are the same to ensure the codec conformance.

This method increases the memory size required on the encoding side to 175% because there is no recompression process and more reference blocks take up more memory. On the decoding side, the use of memory as shown in Fig. 5, remains unchanged, which is equal to the original IBC buffer size in VVC. The method without Cache greatly improves the coding speed and coding effect. It is a very efficient solution in some cases where you have enough resources for the encoder and want to control the cost of the decoder, e.g. cloud gaming and cloud meeting.

The introduction of CurrBUF simplifies the compression process of reference blocks. The arrangement of compression blocks conflicts with the division of CU but the arrangement of VPDUs does not. Therefore, compression after storing a VPDU is easier to perform. Besides, samples stored in CurrBUF can also be used as references for IBC, which may also allow the encoder to find a better reference block, comparing to the block that may be quantized in BUF.

The proposed buffer with RBRC has a fixed size and a mapping rule to find the required reference block from BUF:

$$RefX\_Buf = RefX\_Pic \ \% \ BufWidth \tag{1}$$

$$RefY\_Buf = RefY\_Pic \ \% \ BufHeight \tag{2}$$

BufHeight equals the CTU size and BufWidth equals the memory size of BUF part divided by Buf_Height. Codec uses them to convert the reference block position in picture coordinates (RefX_Pic, RefY_Pic) to the reference block position in BUF coordinates (RefX_Buf, RefY_Buf) and get the required block in BUF. When getting a reference block, whether it is obtained from CurrBUF or BUF also needs to be judged.

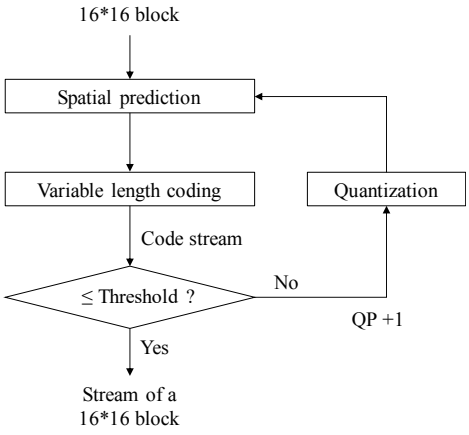

Fig. 6. The compression process of a compression block

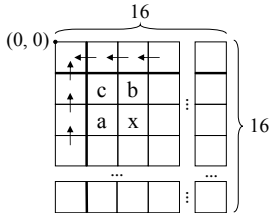

Fig. 7. The Spatial Prediction method

B. Lossy Reference Block Recompression

The Reference Block Recompression (RBRC) is a new recompression method for IBC [6]. Quantization is introduced to change the RBRC into a lossy recompression method. The whole process is divided into three steps: spatial prediction, variable length coding, quantification. The working flow is shown in Fig. 6.

The size of a compression block is set to 16*16, a compression block is compressed with a compression rate of no more than 50% and is stored in a fixed location with half of the original memory space, showing in Fig. 2. The decompression process is the inverse process of compression. Compressed blocks can achieve pixel-level random access by decompressing all relative blocks in the corresponding position. When the side length of the picture isn't divisible by the side length of compression block, the VPDU that crosses the corresponding picture boundary will be removed from the reference area of IBC.

1) Spatial Prediction: We use the prediction method in [21], which has both low complexity and good effect. The block to be compressed is divided into three parts: the top-left sample, the first row and column, and the remaining samples, showing in Fig. 7. The top-left sample is stored without compressing as reference for others; every sample in the first row use the sample on the left as a reference; every sample in the first column use the sample on the top as a reference; the remaining samples are predictions by the samples on the left, top and top-left, respectively represented by a, b and c:

TABLE I. SMALL-VALUE OPTIMIZED VLC TABLE

Table Code	00	01	10	110
Max Value	0	1	2	3~4
0	-	1	01	001
±1		0S	1S	01S
±2			00S	10S
±3				11S
±4				000S
Table Code	1110	11110	111110	111111
Max Value	5~8	9~16	17~32	>32
0	0001	00001	00001	
±1	001S	0001S	0001S	
±2	010S	0010S	0010S	
…	…	…	…	
±7	111S	0111S	0111S	
±8	0000S	1000S	1000S	
…		…	…	
±11		1011S	1011S	xx…xS
±12		1100S	11000S	
…		…	…	
±15		1111S	11011S	
±16		00000S	1110000S	
…			…	
±31			1111111S	
±32			0000000S	

S indicates the sign of the residual.

TABLE II. EXPERIMENTAL RESULT PROPOSED METHOD VS VTM10.0

	Sequence	Proposed Method with Cache					Proposed Method without Cache				
		BD-rate (%)			*Enc. Time*	*Dec. Time*	BD-rate (%)			*Enc. Time*	*Dec. Time*
		Y	*U*	*V*			*Y*	*U*	*V*		
Class F	ArenaOfValor	0.03	0.08	0.02	386%	130%	-0.01	0.09	-0.04	110%	127%
	BasketballDrill Text	0.00	0.07	-0.09	431%	161%	-0.06	-0.09	-0.15	106%	142%
	SlideEditing	-3.64	-3.70	-3.84	397%	296%	-5.96	-5.92	-6.11	102%	140%
	SlideShow	-2.02	-2.05	-1.63	374%	162%	-4.52	-4.48	-4.50	106%	132%
	Average	**-1.41**	**-1.40**	**-1.39**	**397%**	**187%**	**-2.64**	**-2.60**	**-2.70**	**106%**	**135%**
Class TGM	ChineseEditing	-0.51	-0.50	-0.52	379%	370%	-2.15	-2.13	-2.10	102%	148%
	Console	-1.42	-1.48	-1.45	185%	261%	-2.76	-2.78	-2.79	93%	128%
	Desktop	-3.76	-3.93	-3.95	206%	285%	-8.00	-8.03	-8.06	92%	130%
	FlyingGraphics	-1.21	-1.49	-1.43	252%	286%	-2.94	-3.23	-3.18	104%	129%
	Average	**-1.72**	**-1.85**	**-1.84**	**256%**	**301%**	**-3.96**	**-4.04**	**-4.03**	**98%**	**134%**

$$x = \begin{cases} \min(a,b), & if\ c \geq \max(a,b) \\ \max(a,b), & if\ c \leq \min(a,b) \\ a+b-c, & \text{otherwise} \end{cases} \quad (3)$$

Through the prediction process, the block is transformed into a block of residual (except for the top-left sample, it remains its original value).

2) Variable Length Coding: After spatial prediction, we get the block of residual instead of the origin block, and we use the way of variable length coding (VLC) to turn it into a stream to further compress it. The adjusted Small-Value Optimized Variable Length Coding (SVO-VLC) [17] is showing in Table I. The maximum value of the block is count to get a table code, and values of each position on the block are converted into corresponding code according to the table code. A block of residual can be further divided into four small VLC coding blocks to get better coding efficiency, we judge whether to further divide by whether there is a small block with table code of 00 or 01.

3) Quantization: Quantization helps to make sure that the compression rate does not exceed 50% to meet our memory allocation for compressed blocks. The process of quantization and compression is shown in Fig. 6. The quantization parameter (QP) represents the number of bits, by which the value of the sample will be shifted right during the spatial prediction process, with the initial value of zero.

$$Threshold = BlockSize \times PixelBitDepth \times 50\% \quad (4)$$

If the total number of bits in the stream exceeds the threshold, it will go back to the spatial prediction state and quantify the block by shifting right the original value of samples. Every time it exceeds the threshold, it shifts one more bit. There are three bits to present how many bits the block has shifted, so the max bits that can be shifted is 7. If the total number of bits still exceeds the threshold after shifting 7 bits, the output stream of this block will be set to zero to prevent it from being referenced by IBC CU.

III. EXPERIMENTAL RESULT

The experiment is carried on all the 8 YUV420 test sequences specified in VVC common test conditions for SCC [22, 23] under the configuration of All Intra, and in the mode that IBC, HashME, BDPCM is turned on. Our algorithm is embedded in VTM 10.0 [24], the reference software of VVC. The experimental result is compared with the VTM 10.0 with its original IBC algorithm under the same configuration.

The experimental result in TABLE II shows that our proposed method with cache can achieve 1.8% BD-rate reduction on average for the TGM sequences. In the sequences of ArenaOfValor and BasketballDrillText, the BD-rate increases 0.08% in the worst case. The reason is that IBC doesn't work well on these contents [1, 25], these two sequences are game screens and nature contents respectively, there are less repeat patterns, causing no effect to expand the reference area of IBC. In terms of time, the coding time is fluctuating, with encoding time from 185% to 431% and decoding time from 130% to 296%, greatly affected by the cache efficiency. The increase of encoding time is because the complexity of getting reference blocks for IBC increases, and a larger reference area extends the motion estimation process of IBC. For the decoding time, it's fluctuation is mainly because of the cache uncertainty, directly decompressing without cache should be a better way to read reference blocks for the decoder.

For the proposed method without cache, it achieves 4.0% BD-rate reduction on average for the TGM sequences. This result comes from the improvement on IBC with a larger reference area (175%). It proves that compressed reference area is still effective for the IBC mode. In the sequences of ArenaOfValor and BasketballDrillText, the BD-rate increases 0.09% in the worst case. The reason is the same as mentioned above. The encoding time is about 100% and decoding time is about 130%, which is worthwhile comparing to the gain obtained.

IV. CONCLUSION

In this paper, a method of constructing an IBC buffer with the Reference Block Recompression (RBRC) technology to improve the effect of IBC is proposed. The new IBC buffer

contains three parts, CurrBUF, BUF, and CACHE, to help to compress and decompress the reference block used by IBC, and two methods are proposed to realize the IBC buffer, namely method with cache and method without cache. The method of constructing IBC buffer is based on the lossy RBRC, the reference block is compressed with a compression rate of no more than 50% by spatial prediction, variable length coding, and quantization, and stored in the proposed buffer in the form of a bitstream. Under the test environment of All Intra, the method with cache expands the reference range of IBC mode by 25% while keeping the IBC buffer size unchanged on both encoding and decoding side. The method without cache expands the reference range of IBC mode by 75% while keeping the IBC buffer size unchanged on decoding side. The experimental result shows that BD-rate reduction of 1.8% and 4.0% on average can be achieved respectively for TGM test sequences.

REFERENCES

[1] X. Xu and S. Liu, "Screen Content Coding in Recently Developed Video Coding Standards," in 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), 2020, pp. 1-2.

[2] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging HEVC screen content coding extension," IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 1, pp. 50-62, 2016.

[3] J. Chen, Y. Ye, and S. H. Kim, "Algorithm description for Versatile Video Coding and Test Model 10 (VTM 10)," in document JVET-S2002, Proc. of 19th JVET Meeting, teleconference, June 2020.

[4] X. Xu, X. Li, and S. Liu, "Intra block copy in Versatile Video Coding with Reference Sample Memory Reuse," in 2019 Picture Coding Symposium (PCS), 2019, pp. 1-5.

[5] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in document VCEG-M33, Proc. of 13th VCEG Meeting, Austin, Texas, USA, Apr. 2001.

[6] J. Hu, J. Wang, G. Zhong, J. Cao, R. Mao, and F. Liang, "A Lossless Intra Reference Block Recompression Scheme for Bandwidth Reduction in HEVC-IBC," in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021: in press.

[7] T. Song and T. Shimamoto, "Reference frame data compression method for H. 264/AVC," IEICE Electronics Express, vol. 4, no. 3, pp. 121-126, 2007.

[8] T. L. B. Yng, B. Lee, and H. Yoo, "A low complexity and lossless frame memory compression for display devices," IEEE Transactions on Consumer Electronics, vol. 54, no. 3, pp. 1453-1458, 2008.

[9] J. Kim and C. Kyung, "A lossless embedded compression using significant bit truncation for HD video coding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 6, pp. 848-860, 2010.

[10] D. Zhou et al., "A 530 mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip," IEEE Journal of Solid-State Circuits, vol. 46, no. 4, pp. 777-788, 2011.

[11] X. Bao, D. Zhou, P. Liu, and S. Goto, "An advanced hierarchical motion estimation scheme with lossless frame recompression and early-level termination for beyond high-definition video coding," IEEE Transactions on Multimedia, vol. 14, no. 2, pp. 237-249, 2012.

[12] X. Lian, Z. Liu, W. Zhou, and Z. Duan, "Lossless frame memory compression using pixel-grain prediction and dynamic order entropy coding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 1, pp. 223-235, 2016.

[13] S. Yoon, S. Jun, Y. Cho, K. Lee, H. Jang, and T. H. Han, "Optimized lossless embedded compression for mobile multimedia applications," Electronics, vol. 9, no. 5, 2020.

[14] D. Pau and R. Sannino, "MPEG-2 decoding with a reduced RAM requisite by ADPCM recompression before storing MPEG-2 decompressed data," United States, 1998.

[15] C. Cheng, P. Tseng, and L. Chen, "Multimode embedded compression codec engine for power-aware video coding system," IEEE Transactions on Circuits and Systems for Video Technology, vol. 19, no. 2, pp. 141-150, 2009.

[16] T. Tsai and Y. Lee, "A 6.4 Gbit/s Embedded Compression Codec for Memory-Efficient Applications on Advanced-HD Specification," IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, no. 10, pp. 1277-1291, 2010.

[17] Y. Fan, Q. Shang, and X. Zeng, "In-block prediction-based mixed lossy and lossless reference frame recompression for next-generation video encoding," IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, no. 1, pp. 112-124, 2015.

[18] L. Guo, D. Zhou, J. Zhou, S. Kimura, and S. Goto, "Lossy Compression for Embedded Computer Vision Systems," IEEE Access, vol. 6, pp. 39385-39397, 2018.

[19] X. Lian, Z. Liu, W. Zhou, and Z. Duan, "Parallel Content-Aware Adaptive Quantization-Oriented Lossy Frame Memory Recompression for HEVC," IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 4, pp. 958-971, 2018.

[20] A. Willème, B. Macq, A. Descampe, and G. Rouvroy, "Power-Aware HEVC Compression Through Asymmetric JPEG XS Frame Buffer Compression," in 2018 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 3598-3602.

[21] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," IEEE Transactions on Image Processing, vol. 9, no. 8, pp. 1309-1324, 2000.

[22] X. Xu, Y.-C. Sun, Y.-H. Chao, and J. Xu, "Description of Core Experiment 8: Screen Content Coding Tools," in document JVET-L1028, Proc. of 12th JVET Meeting, Macao, CN, Oct. 2018.

[23] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations for SDR video," in document JVET-N1010, Proc. of 14th JVET Meeting, Geneva, CH, Mar. 2019.

[24] The VTM reference software for VVC development, version 10.0. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-10.0

[25] Y. Hu, Y. Li, Z. Chen, X. Xu and S. Liu, "Performance Analysis of Intra Block Copy for Screen Content Coding in AVS3," 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Shenzhen, China, 2020, pp. 123-126.