High performance hardware architectures for Intra Block Copy and Palette Coding for HEVC Screen Content Coding extension

Rishan Senanayake, Namitha Liyanage
Sasindu Wijeratne, Sachille Atapattu
Kasun Athukorala, P.M.K. Tharaka
Geethan Karunaratne, R.M.A.U. Senarath
Ishantha Perera, Ashen Ekanayake
Paraqum Technologies
Dehiwala, Sri Lanka
Email: {rd.senanayake, namitha, sasindu, sachille, kasun.athukorala, kasun, geethan, aruna, ishantha, ashen}@paraqum.com

Ajith Pasqual
University of Moratuwa
Moratuwa, Sri Lanka
Email: pasqual@ent.mrt.ac.lk

***Abstract*—Screen content coding (SCC) extension to High Efficiency Video Coding (HEVC) offers substantial compression efficiency over the existing HEVC standard for computer generated content. However, this gain in compression efficiency is achieved at the expense of further computational complexity with several resource hungry coding tools. Hence, extension of SCC to HEVC hardware encoders can be challenging. This paper presents resource efficient hardware designs for two key SCC tools, Intra Block Copy and Palette Coding. Moreover, a new hash search approach is proposed for Intra Block Copy, while a hardware friendly palette indices coding scheme is suggested for Palette Coding. These designs are targeted to achieve the throughput necessary for an 1080p 30 frames/s encoder, and incurs coding loss of 11.4% and 5.1% respectively in all intra configurations. The designs are synthesized for a Virtex-7 VC707 evaluation platform.**

***Index Terms*—high efficiency video coding, screen content coding, real time systems, video coding on fpga, low latency encoding**

I. Introduction

The High Efficiency Video Coding (HEVC) standard was introduced by the Joint Collaborative Team on Video Coding (JCT-VC) as the successor for H.264/MPEG-4 Advanced Video Coding (AVC). Despite the 50% gain in compression efficiency, HEVC, similar to H.264 was developed primarily for camera captured sequences. However, recent video applications employ more than just camera captured material, but also computer generated video such as text and graphics with motion and animation [1]. Therefore, a new extension was initiated in 2014 to achieve better compression for such screen content (SC). This development exploits unique characteristics in SC including large uniformly flat areas, repeated patterns, highly saturated or a limited number of different colors, and numerically identical blocks or regions among a sequence of pictures. SC extension was finalized in February 2016, and undergoes incremental development [2]. The reference encoder for SC, JCTVC Screen Content Model Software Ver. 8 (SCM-8.0) [3] offers over 50% bit-rate reduction across all encoding profiles for text and graphics with motion video content in comparison with HEVC test model, HM reference encoder version 16.7 [4]. Sizable gains are also witnessed in mixed content and animation videos [1].

The new extension for the HEVC standard Rec. ITU-T H.265 |ISO/IEC 23008-2 version 4 [2], aimed at achieving higher compression efficiency for SC, introduces several coding tools for Intra coding in addition to intra prediction. These tools include Intra Block Copy (IBC), Palette Coding (PLT), Adaptive Color Transform and Adaptive Motion Compensation Precision [1]. Out of these, Intra Block Copy employs a motion compensation within the current frame. Effective in dealing with repetitive symbols in a picture, IBC is carried out at prediction unit (PU) level and treated as an inter PU for coding. Palette coding can offer better efficiency than the prediction-then-transform approach, especially when coding high contrast regions with limited colors.

However, additional tools over intra encoding brings further computational burden, onto the already computationally complex compression standard for high throughput real time encoding [5]. IBC introduces substantial computational overhead in order to carry out an exhaustive search over frame information for intra frame motion compensation. PLT also entails further computations including several exhaustive searches to employ a different approach of pixel encoding. Such resource hungry computations for these additional features at each coding unit (CU) level add more complexity for the overall computation of traversing the quad-tree. Reference software runs of the two encoders demonstrate two to three times increase in computational time for SCC (Table: I).

978-1-5090-4825-0/17/$31.00 ©2017 IEEE

TABLE I
PERFORMANCE COMPARISON BETWEEN HM AND SCM

		Average bytes per frame	Average PSNR	Encoding time per frame
ChineseEditing	HM	426124.4	45.42	35.96
	SCM	238301.7	47.20	105.39
MissionControl Clip3	HM	265420.7	46.54	29.95
	SCM	113866.7	47.05	73.20
SC Console	HM	188639.5	49.87	25.18
	SCM	69322.2	55.22	48.67

Fast HEVC implementations on software [6] and hardware [5] [7] employ pre-estimations of candidate modes and pre-processing based quad tree pruning to achieve speed up during encoding. However, IBC and PLT cannot thus be directly extended to these implementations as the new SCC tools require brute force searching and complete traversal of the quad-tree to derive the most efficient encoding. Hence these techniques would require corresponding estimations to limit rigorous search and early CU decisions to employ pruning to fit into existing encoding architectures. Moreover, additional dependencies now occur at rate-distortion (RD) optimization, as the SCC coding tools must also be completed to make the final selection for a particular CU.

IBC implementation in SCC is discussed in detail in Xu et al. [8]. However, this approach could introduce a lot of dependencies to RD optimization and require considerable resources to achieve a comprehensive search. Although some complexity reduction techniques have been evaluated in [9]–[12], IBC would still require substantial resources to be implemented. Likewise, PLT implementation in SCM is presented in Xiu et al [13]. Even though prior research work is scarce for PLT hardware design, several approaches have been proposed for K means clustering on hardware [14] [15]. However, the clustering technique implemented on SCM 8.0 can be implemented on hardware more efficiently.

In this paper, we propose new architectures for IBC and PLT to be integrated to high throughput HEVC encoding architectures. Moreover, the hash search in IBC is parametrized to adjust search range depending on resources available. The novelties discussed in this paper include,

- Three stage architecture for IBC
- Three stage architecture for PLT
- New Hash Table to minimize resource usage by x8 with minimal coding loss
- Palette sorting architecture used in PLT comparisons.
- Alternate coding approach to code PLT syntax in parallel to be used in estimating RD cost.

II. INTRA BLOCK COPY(IBC)

A. Overview of IBC

Intra Block Copy, introduced as a coding tool for screen content coding, is similar to inter prediction. In place of previous frames in inter, IBC uses current picture as the reference frame. However, Block Vector (BV) search for IBC differs considerably from Inter prediction.

The current version of IBC implemented in SCM employs several search methods to maximize coding efficiency.

- Most probable candidate matching - Block matching is performed on a list of most probable BV. The candidates for this list contains spatial neighboring BVs and BVs best matched for previous blocks.
- Horizontal and vertical 1-D search - Search is generally done for a region of 1x2 Coding Tree Units (CTUs), but is expanded for the full frame for Nx2N PU and 16x16 CU.
- 2-D local search - A local region of 1x2 CTUs is searched for 8x8 blocks if luma activity for the current block is beyond a threshold of 168. Luma activity is calculated as the minimum of horizontal and vertical gradients.
- Hash Search - Performed for 8x8 CUs with 2Nx2N PU. A hash value is generated to capture features of 8x8 CUs in order to group similar blocks together. Hash value is a 16 bit value formed from 1.

$$
\begin{aligned}
H = (MSB_3(DC_0) << 13) + (MSB_3(DC_1) << 10) \\
+ (MSB_3(DC_2) << 7) + (MSB_3(DC_3) << 4) \\
+ MSB_4(Grad) \quad (1)
\end{aligned}
$$

Where DC_0, DC_1, DC_2 and DC_3 are the luma intensity averages of sub-divided 4x4 blocks from an 8x8 CU. *Grad* is the average of horizontal luma gradient and vertical luma gradient. $MSB_m(X)$ refers to the *m* most significant bits of *X*. Blocks that can be easily predicted using intra prediction are removed from hash table. Hash table is updated after every CTU and reset after every frame. [1]

Since large CU sizes are less likely to encounter repeating patterns within a frame [16], most of the search is done for 8x8 CUs. 32x32 CU is searched only in most probable candidate search and 16x16 CU in horizontal and vertical search.

B. Overall IBC architecture

The IBC operation is divided into two stages; estimation and high throughput stage. In estimation stage most probable candidates are selected using original pixels of the previously coded area. As the estimation stage is devoid of feedback, it can operate in parallel to the reconstruction dependent loop. High throughput stage create residuals using reduced candidate list. The next subsections describe these sections in detail.

C. Estimation Stage

In the proposed hardware architecture, a single processing element is employed to perform both forms of local searches, horizontal and vertical 1-D local search and the 2-D local search. Figure 1 shows the local search module which can search BV candidates for one 8x8 CU in a single pass. Four such modules are used in parallel to process four 8x8 CUs simultaneously. There are five modules to check BV validity of each candidate PU size. The derived best candidates are used to create combined BV list. Search area for all CU sizes has been limited to 1x2 CTUs in 1-D and 2-D search to optimize the trade-off between timing restrictions and resources. If

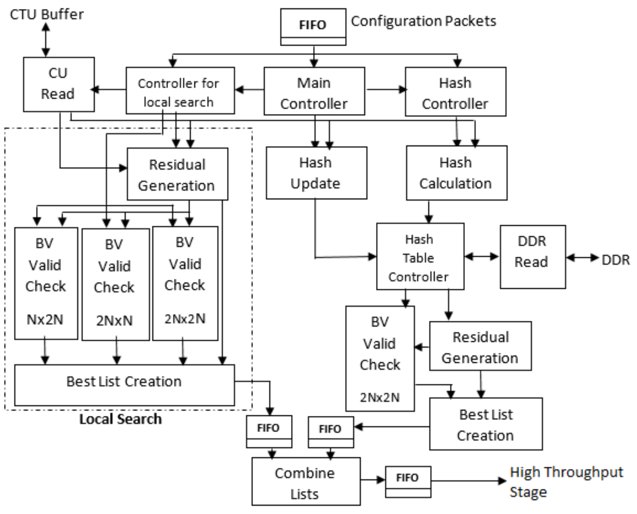

Fig. 1. Block diagram of IBC estimation stage

resources are not a constraint, modules can be added in parallel to current local search module to expand the search area.

The number of BVs stored per hash key is unrestricted in SCM. This has to be limited in hardware although it reduces coding efficiency. When analyzing the distribution of the hash table in SCM, large number of empty bins can be observed. Therefore, it is desirable to uniformly distribute the hash values. To achieve that, 4 bit gradient is reduced to 1 bit making hash key size 13 bits. This reduces hash table size by a factor of 8. The new hash function is shown in 2.

$$H = (MSB_3(DC_0) << 10) + (MSB_3(DC_1) << 7 \\ + (MSB_3(DC_2) << 4) + (MSB_3(DC_3) << 1) + \delta \quad (2)$$

Where δ is set to zero only if four Most Significant Bits of gradient is zero, else δ is set to one.

Table II shows the results for the SCM implemented 16 bit hash search with a limited number of BVs per hash key and proposed 13 bit hash scheme. Incremental coding gain is calculated with respect to SCM implementation excluding hash search. Comparison shows that proposed hash scheme is almost similar as or slightly better than the existing hash scheme when hash table size needs to be constrained.

Hash update module calculates hash keys for CTU and updates the table. Hash table is stored in external DRAM and information about filled status of hash table for each key is stored in block RAMs. Using these information BVs are fetched from the hash table and then predicted blocks are fetched according to BVs. Hash matching process is pipelined so that effect of DRAM latency is reduced.

D. High Throughput Stage

In the high throughput stage, multiple BVs are predicted and evaluated to refine the BVs selected from estimation stage. This stage predicts a CU in five different combinations and choose the best prediction.

TABLE II
BITRATE AGAINST PSNR PERFORMANCE WITH THE MAXIMUM POSSIBLE HASH KEY ITEMS FOR THE EXISTING AND PROPOSED HASH SCHEMES

Maximum candidate limit per hash key	Average Δ BD rate for existing Hash scheme	Average Δ BD rate for proposed Hash scheme
1000	-4.8812	-4.9735
100	-8.5989	-8.7481
50	-10.1337	-10.2859
10	-13.9869	-14.5229

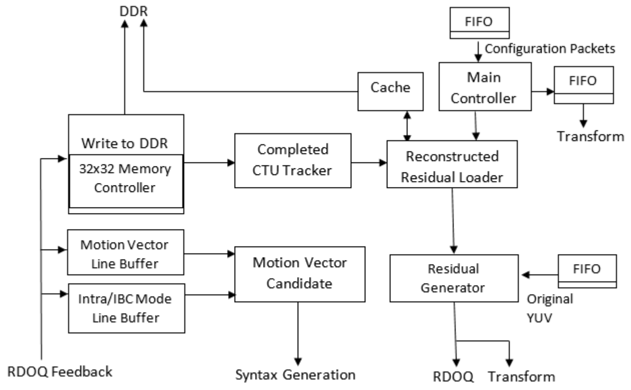

Fig. 2. Block diagram of IBC high throughput stage

Main constraint in the high throughput stage is the memory accesses for external DRAM memory since it has a high memory access latency. As shown in Figure 2, a cache module is used to reduce this latency. The Main Controller reads configuration packets from a FIFO and calculate corresponding reconstructed reference pixel locations for IBC prediction. Generated locations are sent to Reconstructed Residual Loader. It loads the reconstructed reference pixels through the cache module and generate prediction data which are then sent to the Residual Generator for pixel residual value generation.

Reconstructed pixel feedback to Write to DDR module is buffered in a 32x32 memory block. It writes to DRAM in row basis using 256 bit AXI data bus. Result of this module is sent to the Completed CTU Tracker to update the validity of the DRAM memory. Completed CTU Tracker module keeps track of the completion of the reconstructed memory writes along the CTU rows. Parameters required for the Syntax Generator is provided by Motion Vector Candidate module using prediction mode and motion vectors of the neighboring CUs.

Assuming the reconstructed pixels are always available when needed and no cache misses, the timing details can be derived for the given architecture for producing five different predictions for a CU. Table III shows timing details derived for a 1080p video at 30 frames/s.

III. PALETTE CODING (PLT)

A. Overview of Palette Coding

Palette coding is a new tool used in HEVC-SCC to efficiently code areas having few distinct colors. Unlike camera captured content which has continuous shades of colors,

TABLE III
TIMING INFORMATION FOR IBC HIGH THROUGHPUT STAGE

CU size	8x8	16x16	32x32
Available clock cycles	140	560	2240
Required clock cycles	45	125	475

computer generated content have highly contrasting regions of uniform colors [1]. When transformation based conventional approach is used to code such regions, edges get distorted or blurred due to clipping of high frequency components in the transformation process.

The proposed palette coding approach used in HEVC-SCC indicates the color values of each pixel separately. The different color values are listed in a table called palette prediction table and each pixel is assigned an index representing the location of its color value in the list. The standard has provisions to denote a sequence of similarly indexed pixels efficiently, reducing the number of bits needed to transmit. Moreover, only the new entries of the palette prediction table are sent to the decoder and existing entries are simply signaled whether they are used in the current block or not.

B. Hardware Implementation

A palette coding algorithm is implemented in the reference software encoder (SCM-8.0) [3] with three main stages [13]. They are palette table derivation, palette table coding and palette index coding. In palette table derivation, first the histogram of each CU is clustered around few initial points. This process is iterated few times to find the best center points of clusters for the given block. In palette table coding each center point is compared with the nearest entry in palette table of previous block to decide whether to code as a separate entry. Palette index coding is then performed to find the cost of the CU palette operation. This process starts from the largest CU and continues downward the quad tree structure.

This approach however makes real time palette coding a challenge. As coding of one CU block depends on the completion of previous palette coding block, the latency of the operation is more critical than throughput. Moreover, the iterative optimization explored in software may not be as attractive in hardware implementations. Therefore, in this section we have introduced a new palette coding architecture by reducing the operations depending on feedback by pre-processing and using estimations.

C. Overall Palette Architecture

The proposed architecture consists of three stages; palette clustering operation, palette prediction list and palette indices list creation and palette entropy coding stage. Among the above three stages only the palette prediction list and palette indices list creation stage is dependent on feedback making it the only stage needed to be optimized for latency.

D. Palette Clustering Operation

Clustering requires considerable amount of resources and significant time for the operation making real time encoding a challenge. Therefore, we use an initial clustering based only on raw pixel values. Hence the complex operation of palette clustering is removed from the feedback dependent path.

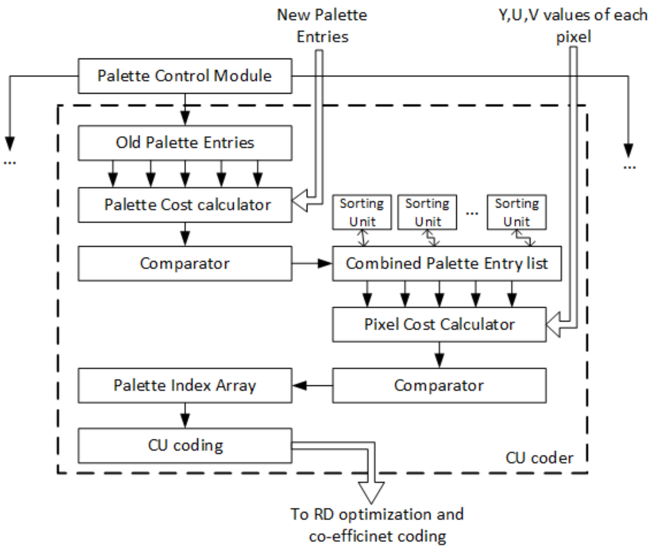

Fig. 3. Architecture of palette prediction list and palette indices list generation module

Raw YUV values of each 8X8 CU is fed into the palette clustering module. Histogram based refinement of K-mean clustering algorithm is used to cluster the pixels. The error margin is derived from QP using the same equation as used in SCM [3]. By combining the results of successive blocks the means of clusters of corresponding 16x16 and 32x32 blocks can be obtained. Throughput is increased by pipelining these stages. The number of maximum clusters are limited to 64 to limit the resource usage. The resulting slight increase in distortion is justified by the gains in both time and resource utilization.

E. Palette Prediction List and Palette Indices List Generation

Figure 3 shows the architecture of this module consisting of palette controller module and 3 CU coders for 8X8, 16X16 and 32X32 CUs. In figure 3, only one CU coder is described in detail as others also have a similar architecture. As this module is feedback dependent, more priority is given to reduce the latency of the operation.

1) Palette controller module: This module acts as the central controller of the palette operations. It analyzes the quad tree partition structures used in other prediction channels for the same block (intra and IBC) and the selected prediction mode sizes of previous blocks. From these data it schedules the process order of CU coders. Depending on the CU size that is being processed in intra and IBC, more than one CU can be processed serially without waiting for the feedback. This is a major advantage of this parallel processing CU Coding system. In addition, the controller module maintains most recently used palette predictor table of each CU coder and updates them depending on the final selection from RD optimization.

2) CU coders: Each CU coder accepts the list of centers of the clusters in the relevant CU, one item at a time from the palette clustering module. First CU coder maps each incoming such palette with existing palette items to find the closest match. Absolute difference between the new palette and each old palette is used as the metric to calculate the closest match. A three stage pipelined comparator system is implemented to find this match. The incoming palette is then added to the palette list if the sum of squared difference with the closest match is greater than a predefined threshold.

As the new palettes are fed into the above pipe-lined system sequentially, if n number of new palettes are fed into the system, modified palette list creating process is completed in $n+6$ clock cycles. In comparison, the SCM encoder compares each old palette entry with a list of new palette entries to find the closest match. If this procedure is used in hardware the palette mapping process cannot be started until the transfer of all new palette entries have been completed. Therefore, by using our approach we were able to eliminate a significant idle time in a latency critical module.

The drawback in this approach is that the selected old palette entries and new palette entries get interleaved. As old palette indices must be followed by new-palette indices, a palette sorting module is added to our architecture. 32 2-element comparators compare each element at even positions in the palette array with the element above and element below in alternative clock cycles.

After the palette table is created each pixel value is compared with combined palette entry list to find the closest palette entry similar to the one used to map old palette entries with new palette entries. As there are 64 pixels, the process is sped up by dividing the pixel array into four sub blocks at the expense of resource usage.

The final step in the CU coder is the coding of palette indices by properly utilizaing 'copy from above', 'copy from left' and 'copy from above final' flags. To prevent missing of 'copy from above final flag', each segment is coded in reverse so that preceding pixel of the first occurrence of copy from left flag is marked as copy from above final flag. The selection of 'copy from above' and 'copy from left' is done by entropy coding in SCM. However, this approach hinders parallelism and it is estimated in our approach by selecting the mode which has the longest run length. In order to speed up the coding process, each CU is divided into 4 and combined by connecting flags in the boundaries.

In SCM encoder, the data generated from palette coding is entropy coded to obtain the number of bits used for palette coding to compare it with other prediction directions. However, this operation is time consuming and not suitable for a feedback constrained loop. Therefore, we estimate the number of bits used for each of these operations using bits needed for binary conversion. The drawback in this approach is that the bits reduced form Context Adaptive Binary Arithmetic Coder (CABAC) is not considered. However, this can be justified, as eliminating the CABAC enables to calculate the number of bits for above syntax elements in parallel.

F. Palette entropy coding stage

The final stage in PLT consists of entropy coding the selected old palette entries, new palette entries, CU Coding data (traversing method, traversing count, palette index) and escape flags. As feedback of this stage is not needed, a multi-stage pipeline is implemented to gain the required throughput in entropy coding similar to the one used in [5].

IV. Results

Our proposed modifications for IBC and PLT were implemented on reference HEVC encoder SCM 8.0 [3]. Results for Screen content, Animation and for mixed content video sequences are provided in Table IV with reference to intra-main-scc profile. Proposed changes in palette mode results BD rate loss of only 5.10% on average. Proposed changes related to IBC with SCM 8.0 implemented hash search results 11.39% BD rate loss and IBC changes with the new hash for 100 BVs per hash key incur a BD-rate penalty of 18.55% in average. These BD rate reductions are compensated with improved throughput and lower memory requirement (memory requirement drops to 1/8th of that of the HM). BD plots for several video sequences are provided in Figure 4 and Figure 5 for proposed IBC and palette changes. From these graphs it is clearly visible that the BD rate drop with respect to the reference HM-intra-main-scc encoder by the proposed architecture is justifiable compared to the BD rate gain relative to that of the HM-intra-main encoder. The proposed architecture provides the opportunity of hardware implementation of SCC modules with real time operation compromising coding efficiency.

The proposed architecture for HEVC-SCC modules is targeting Xilinx VC707 platform. The design is targeted to handle an average throughput of 0.45 pixels per clock cycle, sufficient to support 1080p 30 frames/s video. Individual modules are initially synthesized at 200MHz during standalone implementation and the integrated design is synthesized at 140MHz.

Resource utilization for key modules are listed in the Table V. PLT unit requires a total of 66K LUTs and it has to be incorporated to an existing HEVC encoder as a separate unit. IBC unit takes 51K LUTs for an standalone implementation but an existing inter prediction architecture can be upgraded to support IBC with only 38K LUTs by resource sharing with inter prediction units.

V. Conclusion

This paper suggests algorithms and architectures for hardware implementation of SCC IBC and PLT tools. Additionally, several novel approaches are proposed for these tools to achieve better resource utilization. However, a coding overhead is introduced for each coding tool, estimated to be 11.4% and 5.1% respectively. In spite of that, this trade-off should enable real time 30 frames/s encoding in 1080p. Future extensions of this project may include hardware designs for additional SCC coding tools and integration of these SCC coding tools to develop a real time SCC encoder on hardware.

TABLE IV
ESTIMATED INCREASE OF BIT-RATE AGAINST PSNR FOR PROPOSED HARDWARE DESIGN FOR SCC CODING TOOLS

Sequence	Resolution	Δ BD-rate %		
		PLT	IBC	IBC new hash
Screen content				
ChineseEditing	1920x1080	8.60	5.09	10.66
sc_console	1920x1080	3.67	24.48	28.65
sc_desktop	1920x1080	3.86	16.52	23.02
sc_map	1280x720	3.15	2.39	3.98
Animation				
sc_robot	1280x720	0.05	0.52	0.65
ChinaSpeed	1024x768	3.6633	17.59	35.42
Mixed				
MissionControl	1920x1080	3.02	7.02	17.52

TABLE V
RESOURCE UTILIZATION OF PROPOSED UNITS

Module	LUT	REG	Block RAM
Palette coding unit			
Palette Clustering	20K	5K	8
Palette Controller	20K	5K	-
CU Coder	12K	2K	-
Palette Coding	14K	3K	14
Intra block copy unit			
2D Search	23K	8K	3
HASH Search	15K	4K	3
High Throughput Stage	13K	11K	24

REFERENCES

[1] 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.
[2] R. Joshi, S. Liu, J. Xu, G. Sullivan, G. Tech, Y. Wang, Y. Ye *et al.*, "High efficiency video coding (hevc) screen content coding: Draft 6," *Document of Joint Collaborative Team on Video Coding, JCTVC-W1005-v4*, 2016.
[3] "Scm software repository at hevc screen content coding extension (scc)," https://hevc.hhi.fraunhofer.de/scc/, 2016, [Online; accessed April 2017].
[4] "HM software repository," accessed 2016. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/

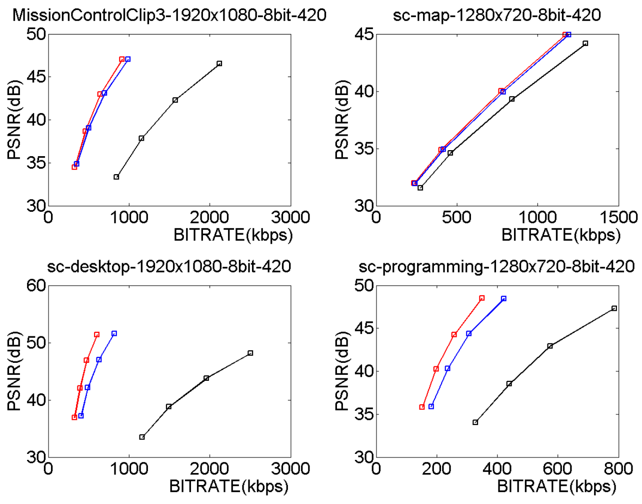

Fig. 4. Comparison of bit rate against PSNR of SCM reference encoder(red), proposed IBC design(blue) and HM encoder(black)

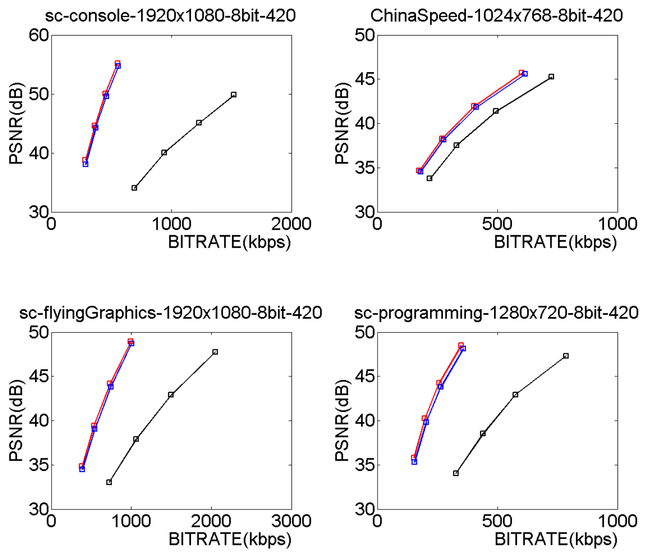

Fig. 5. Comparison of bit rate against PSNR of SCM reference encoder(red), proposed PLT design(blue) and HM encoder(black)

[5] S. Atapattu, N. Liyanage, N. Menuka, I. Perera, and A. Pasqual, "Real time all intra hevc hd encoder on fpga," in *Application-specific Systems, Architectures and Processors (ASAP), 2016 IEEE 27th International Conference on*. IEEE, 2016, pp. 191–195.
[6] L. Zhao, L. Zhang, S. Ma, and D. Zhao, "Fast mode decision algorithm for intra prediction in hevc," in *Visual Communications and Image Processing (VCIP), 2011 IEEE*. IEEE, 2011, pp. 1–4.
[7] G. Pastuszak and A. Abramowski, "Algorithm and architecture design of the h. 265/hevc intra encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 210–222, 2016.
[8] X. Xu, S. Liu, T.-D. Chuang, Y.-W. Huang, S.-M. Lei, K. Rapaka, C. Pang, V. Seregin, Y.-K. Wang, and M. Karczewicz, "Intra block copy in hevc screen content coding extensions," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 4, pp. 409–419, 2016.
[9] J. Ma, Y.-J. Ahn, and D. Sim, "Intra block copy analysis to improve coding efficiency for hevc screen content coding," *Journal of Broadcast Engineering*, vol. 20, no. 1, pp. 57–67, 2015.
[10] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Hash based fast local search for intra block copy (intrabc) mode in hevc screen content coding," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*. IEEE, 2015, pp. 396–400.
[11] C.-W. Kuo, H.-M. Hang, and C.-L. Chien, "Intra block copy hash reduction for hevc screen content coding," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2016 Asia-Pacific*. IEEE, 2016, pp. 1–9.
[12] S.-H. Tsang, W. Kuang, Y.-L. Chan, and W.-C. Siu, "Fast hevc screen content coding by skipping unnecessary checking of intra block copy mode based on cu activity and gradient," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2016 Asia-Pacific*. IEEE, 2016, pp. 1–5.
[13] X. Xiu, Y. He, R. Joshi, M. Karczewicz, P. Onno, C. Gisquet, and G. Laroche, "Palette-based coding in the screen content coding extension of the hevc standard," in *Data Compression Conference (DCC), 2015*. IEEE, 2015, pp. 253–262.
[14] T. Saegusa and T. Maruyama, "Real-time segmentation of color images based on the k-means clustering on fpga," in *2007 International Conference on Field-Programmable Technology*, Dec 2007, pp. 329–332.
[15] F. Winterstein, S. Bayliss, and G. A. Constantinides, "Fpga-based k-means clustering using tree-based data structures," in *2013 23rd International Conference on Field programmable Logic and Applications*, Sept 2013, pp. 1–6.
[16] D.-K. Kwon and M. Budagavi, "Fast intra block copy (intrabc) search for hevc screen content coding," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 9–12.