In-Block Prediction-Based Mixed Lossy and Lossless Reference Frame Recompression for Next-Generation Video Encoding

Yibo Fan, *Member, IEEE*, Qing Shang, and Xiaoyang Zeng, *Member, IEEE*

***Abstract*— Frame recompression is an efficient way to reduce the huge bandwidth of external memory for video encoder, especially for P/B frame compression. A novel algorithm, which is called mixed lossy and lossless (MLL) reference frame recompression, is proposed in this paper. The bandwidth reduction comes from two sources in our scheme, which differs from its previous designs and achieves a much higher compression ratio. First, it comes from pixel truncation. We use truncated pixels (PR) for integer motion estimation (IME) and acquire truncated residuals for factional motion estimation (FME) and motion compensation (MC). Because the pixel access of IME is much larger than FME and MC, it saves about 37.5% bandwidth under 3-b truncation. Second, embedded compression of PR helps to further reduce data. The truncated pixels in the first stage greatly help to achieve a higher compression ratio than current designs. From our experiments, 3-b truncated PR can be compressed to 15.4% of the original data size, while most current embedded compressions can only achieve around 50%. For PR compression, two methods are proposed: in-block prediction and small-value optimized variable length coding. With these experiments, the total bandwidth can be reduced to 25.5%. Our proposed MLL is hardware/software friendly and also fast IME algorithm friendly frame recompression scheme. It is more suitable to work together with the data-reuse strategy than the previous schemes, and the video quality degradation is controllable and negligible.**

***Index Terms*— Frame recompression, high-efficiency video coding (HEVC), mixed lossy and lossless (MLL), motion estimation, video encoder.**

I. Introduction

AS THE next-generation standard of video coding, High Efficiency Video Coding (HEVC) [1] is intended to provide super high definition (HD), which offers significantly enhanced visual experience. Currently, 1080p HD has already become a mainstream standard for various video applications. Higher specifications such as 4 K × 2 K quad full high definition (QFHD) format, which delivers at least four times the data throughput of HD, have been targeted by next-generation applications. However, the corresponding huge external memory access challenges the design of real-time video encoder very large-scale integration.

The main source of external memory access comes from motion estimation (ME). For 1080p at 30 frames/s FHD video coding, the bandwidth requirement for integer motion estimation (IME) could be as high as 40 Gb/s, if it set the search range to be (−64, 63). For fractional motion estimation (FME) and motion compensation (MC), the bandwidth also can be as high as 3.1 and 1.6 Gb/s. For 4 K video coding with higher frame frequency, the bandwidth requirement could be more than ten times of FHD video coding. Huge external memory bandwidth dramatically increases the difficulty of integrated (IC) design, and also makes the input/output (IO) power consumption unacceptable.

There are three main approaches to reduce the external bandwidth of video encoder. First, fast ME algorithm reduces the search points in search range, such as 4SS [2] or TZ [3] algorithm can reduce at most more than 90% search points. As a fast algorithm makes pixel data access random and unpredictable, it is more suitable for CPU- or DSP-based soft video encoder. For hardware encoder, some other approaches are proposed, such as pixel truncation [17] and hierarchical ME [9]. Secondly, many on-chip data reuse methods are proposed for ME. Tuan *et al.* [4] summarized data-reuse schemes as four levels, levels A–D. The main idea of data reuse is to reduce external memory access through increasing on-chip memory buffer. For example, the level D data reuse needs to store several lines of pixel in frame width on chip, and it can eliminate all of redundant memory access. However, it is too expensive to store lines of pixel on chip for FHD or 4 K video. The third approach is frame recompression. It is a technique to reduce the reference frame size by compressing the original pixel data before such data are stored into off-chip memory. With a decreased amount of fetched data, the bandwidth requirement is reduced as a consequence. The compression algorithm benefits from its fast, high compression efficiency, and ultralow loss or lossless quality.

The frame recompression techniques have been widely studied in the past decade. The transform-based approach is proposed in [5]. However, this approach requires a large amount of computations which is not suitable for low-latency and low-power consumption systems. A down sampling-based recompression method is proposed in [6]. However, the PSNR

Manuscript received July 31, 2013; revised December 1, 2013 and February 27, 2014; accepted May 28, 2014. Date of publication June 5, 2014; date of current version January 5, 2015. This work was supported in part by National Natural Science Foundation of China under Grant 61306023, in part by the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20120071120021, in part by Science and Technology Commission of Shanghai Municipality under Grant 13511503400, and in part by the National High Technology Research and Development (863 Program) under Grant 2012AA012001. This paper was recommended by Associate Editor S.-Y. Chien.

The authors are with State Key Laboratory of ASIC and System, Fudan University, Shanghai 200433, China (e-mail: fanyibo@fudan.edu.cn; xyzeng@fudan.edu.cn).

Digital Object Identifier 10.1109/TCSVT.2014.2329353

may be degraded owing to lossy reference frame. In addition, the differential pulse code modulation (DPCM) is a widely used spatial domain compression method [7]–[10]. It achieves high compression efficiency by reducing the spatial redundancies of image. Many domain-specific methods are also proposed, for example, a lossless frame recompression method for hierarchical ME scheme is presented in [9], but this method can only be applied to some special ME fast search algorithms. TZ search and some other popular fast algorithms cannot be supported by this scheme.

Most of the existing frame recompression methods are designed independently of video encoder. Unfortunately, they did not intend to finely cooperate with encoder, but only focus on improving the compression ratio. Actually, many information can be used for frame recompression and make it work more efficiently with encoder. For example, IME, FME, and MC have different workload and bandwidth requirement, the precision of compression can also be different for each of them. What is more, intra prediction in video encoder can be used to guide the recompression of each coding blocks. Frame recompression should take all of them into consideration, and make use of information from encoder. In this paper, an encoder friendly frame recompression scheme is proposed. This scheme fully uses the information from encoder and discriminate IME from FME and MC to finely save external memory bandwidth. There are two coding layer in our scheme: base layer (BL) and enhancement layer (EL). Base layer is lossy data which only used for IME. Combining with enhancement layer, lossless data can be reconstructed and it is used for FME and MC. Base layer is compressed by proposed three new techniques: tailing-bit truncation (TBT), in-block prediction (IBP), and small-value optimized variable length coding (SVO-VLC).

The rest of this paper is organized as follows. In Section II, frame recompression is briefly introduced. The data-dependency and current solutions are presented. Section III describes the proposed frame recompression scheme. In Section IV, the compression ratio of proposed scheme is evaluated and compared with other methods. Section V gives the implementation of video encoder with proposed mixed lossy and lossless (MLL) scheme, and evaluates the video quality under multibit TBT. Finally, the conclusion is presented in Section VI.

II. Frame Recompression and Motivation

A. Introduction

Frame recompression can be used in both of video encoding and decoding, as shown in Fig. 1. For video encoding, many external data should be loaded on-chip, such as current macroblock (CMB) for coding, reference pixels for IME, FME and MC from reference frames, reference pixels for intra, DB/SAO from current frame and filtered CMB pixels. Similarly, video decoding also needs such kind of external data except for IME and FME.

The right part of Fig. 1 shows a quantitative comparison of bandwidth requirement for video encoding and decoding. If set the basic coding unit, which corresponding to MB in H.264 or CU in HEVC, to be 16×16, and search range to be $(-128, 127)$, the bandwidth requirement for each modules in video codec are dramatically different. IME occupies most of the bandwidth, while MC and intra are very small. Obviously, video encoding faces much more critical bandwidth problem than video decoding.

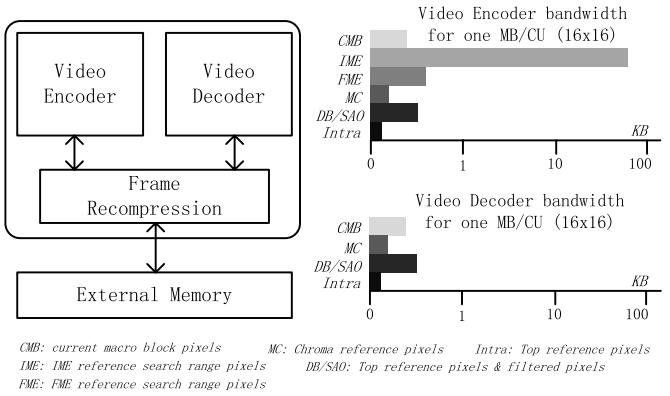


Fig. 1. Frame recompression in video codec.

B. Current Frame Recompression Schemes

1) ADPCM: DPCM is a common signal encoder that uses the baseline of PCM but adds some functions based on the prediction of the samples of the signal. In video coding, DPCM is a spatial domain compression method, which calculates the difference between successively scanned data and uses the difference to represent the data as

$$\mathrm{Res}_n = P_n - P_{n-1} \tag{1}$$

where P_n is the nth pixel, the Res_n is the difference of P_n and P_{n-1}. The DPCM is performed on the 1-D data. However, image is 2-D. Before DPCM, the pixels are lined from 2-D to 1-D by scanning. The scan pattern can be various. For an efficient compression by DPCM, the difference between successive data must be small so that the data can be represented by small number of bits. Therefore, DPCM is normally used along with quantization and VLC. As it uses quantization, the compression becomes lossy in most of cases.

ADPCM [10] is a firstly reported frame compression method which adopts DPCM for video. The algorithm flowchart is shown in Fig. 2(a). The adaptive comes from the quantization coefficient which is adaptively changed according to content of each coding block. The basic coding block for ADPCM is 8×8, and the scan order is a simple raster scan as shown in Fig. 2(a).

2) Intra Mode Referenced ADPCM: As the scan order of ADPCM is very simple, and it cannot fully use the redundant of neighboring pixels, a scan pattern-alterable DPCM scheme is proposed in [7]. It makes use of the information from H.264 intra prediction results. As the intra prediction mode gives the directional information of pixel changes, the DPCM scan order can be adaptively changed according to image contents, which is shown in Fig. 2(b). The basic coding block is 4×4. They proposed eight scan orders for each coding block.

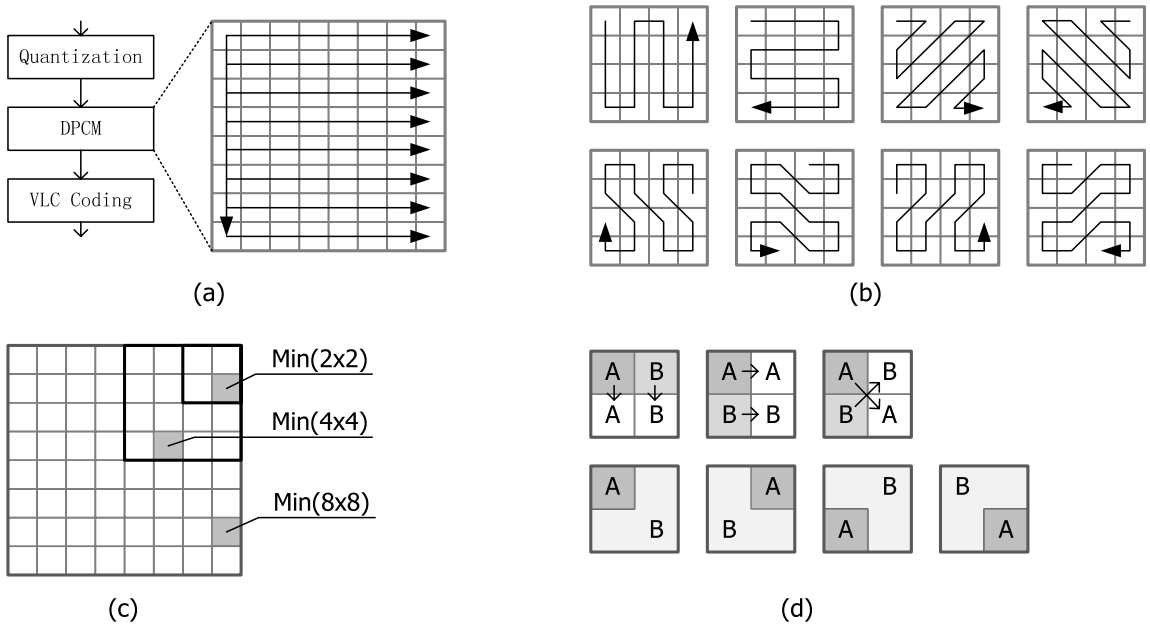


Fig. 2. Existing fame recompression schemes. (a) ADPCM. (b) Intra mode referenced ADPCM. (c) HDM. (d) ERP.

A proper scan order can fully make use of redundancy of neighboring pixels, and produces smaller residuals, which contributes shorter bit length in VLC coding. Golomb–Rice coding is used in their scheme. Many quantization coefficients are tested to determine an optimized one for each coding unit. Finally, this scheme achieves the compression ratio of 50%, but the average PSNR degradation is 1.03 dB. The PSNR degradation mainly comes from the quantization before DPCM.

3) Hierarchical Minimum and Difference: Hierarchical minimum and difference (HMD), which removes quantization and just uses addition operation in decompression process, is proposed for lossless frame recompression in [11]. The basic coding unit for HMD is 2×2. As shown in Fig. 2(c), the minimum pixel is first searched within each 2×2 block, which is called as min2×2. Then the difference between each pixel and the corresponding min2×2 is calculated, which is represented as diffPixel. Second, for each 4×4 block, there are four min2×2, the minimum value among the four $\times$ 2s is found to be called min4×4. The difference between the min2×2 of each 2×2 block and the relevant min4×4 is computed to be called diff2×2. This method is repeated for next bigger blocks. Fig. 2(c) shows an HMD for 8×8 block. The diff4×4, diff2×2, and diffPixel are generated according to min8×8, min4×4, and min2×2. Finally, VLC and packing are performed to compress the min and diff.

In decompression process, the pixel is reconstructed by adding up the diff with min as

$$\begin{aligned} \text{pixel}[a][b][c] &= \text{min}8 \times 8 + \text{diff} \\ &= \text{min}8 \times 8 + \text{diff}4 \times 4[a] \\ &\quad + \text{diff}2 \times 2[a][b] + \text{diffPixel}[a][b][c] \end{aligned} \quad (2)$$

where a, b, and c are the index for 4×4 block, 2×2 block, and pixel, respectively.

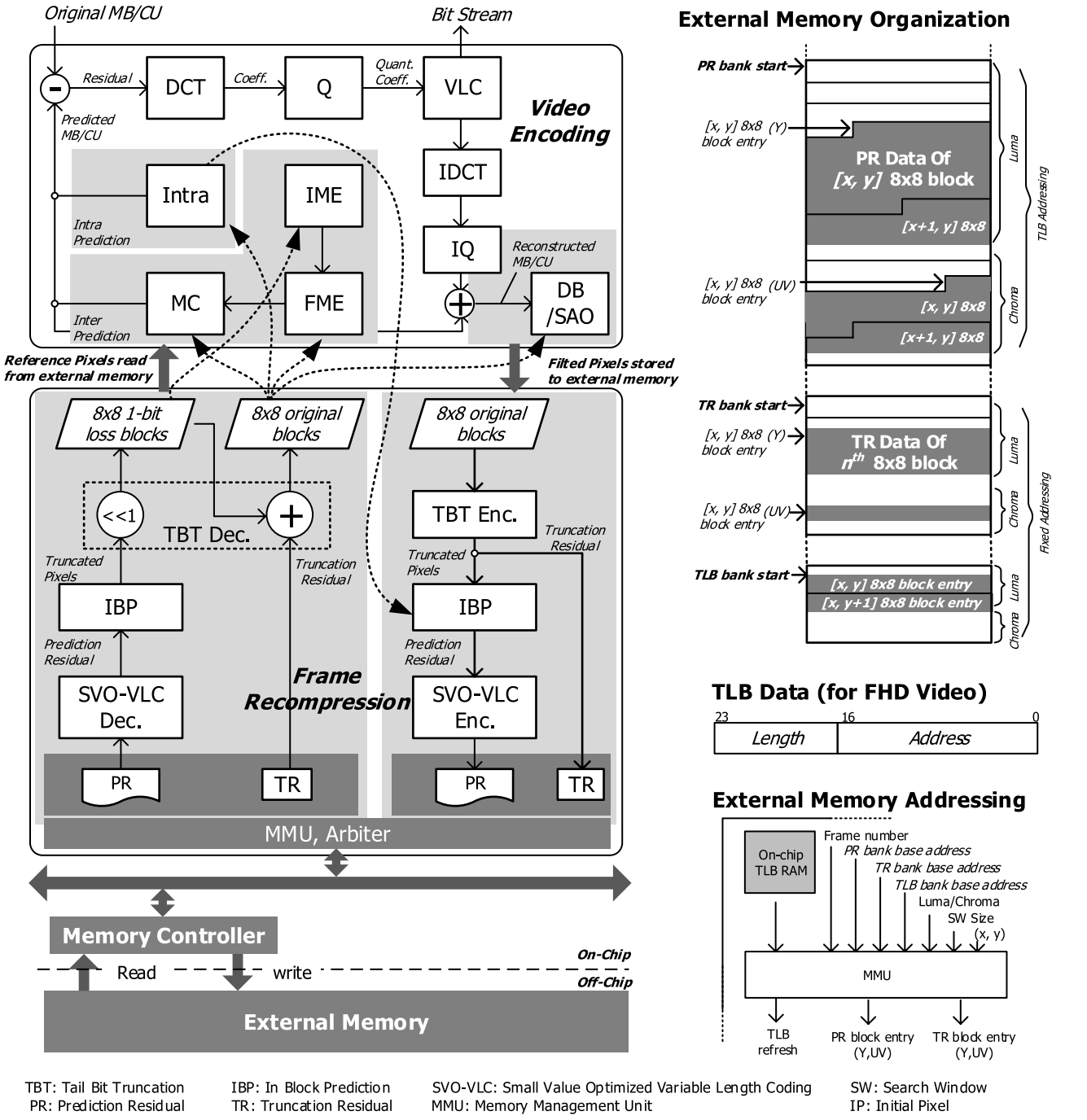
4) Embedded Reconstruction Patterns: Embedded reconstruction patterns (ERP) [12] comprises a predictive pattern decision for encoding of original 2×2 pixel blocks, selective step quantization and fast reconstruction of embedded patterns during decoding. In every 2×2 pixel block, only 2 pixels are encoded as predictors while the other two pixels are predicted by the predictors. Seven reconstruction patterns are proposed in the ERP which is shown in Fig. 2(d). The light gray color illustrates the positions of predictor A and B. After the pattern decision, the selective quantization is performed. For some patterns, it set quantization to 7 b, while for other patters, 6 b quantization is applied. After that, the quantized predictors along with the pattern modes are encoded into 16 b. The EPR can achieve a constant compression rate of 50%, and the compressed data can be randomly accessed. Because the quantization is used, ERP is a kind of lossy compression.

C. Motivation

Most of current pixel compression schemes only achieve about 50% compression ratio, and it seems hard to go a short step. Because the improvement from spatial redundancy of original pixel is very hard, we decided to gain more from truncated pixel.

Pixel truncation has already been used in motion estimation to reduce memory bandwidth [17], and truncated pixel will greatly remove the variations between neighboring points. From our experiments, it can achieve 15.4% compression ratio after 3-b truncation.

On combining pixel truncation and compression, much higher bandwidth reduction than current designs could be can achieved. Especially for video encoder, the truncated pixels can be only used for motion estimation, and then padding the residuals to reconstruct lossless pixels for motion compensation. It causes negligible video quality drop as discussed in [18].



TBT: Tail Bit Truncation IBP: In Block Prediction SVO-VLC: Small Value Optimized Variable Length Coding SW: Search Window
PR: Prediction Residual TR: Truncation Residual MMU: Memory Management Unit IP: Initial Pixel

Fig. 3. MLL frame recompression scheme.

Moreover, current frame recompression schemes mostly are not intended designed to firmly cooperate with video codec. Many useful information from video codec can be used for frame recompression to achieve higher efficiency.

III. Proposed MLL Frame Recompression Scheme

The proposed frame recompression scheme aims to fully cooperate with video encoder to reduce the external memory access and improve the bandwidth utilization. Fig. 3 shows a whole architecture of MLL scheme.

In video encoder part, there are five modules which load reference pixels very frequently: IME, FME, MC, intra, and DB/SAO. DB/SAO stores the final filtered pixels to external memory for future reference. Intra mode from video encoder is used by proposed frame recompression. IME has a distinct source of lossy reference pixels whereas other modules have lossless reference pixels. As IME occupies most of external memory access, lossy pixels for IME expect to reduce more memory bandwidth than previously proposed schemes.

All of memory read/write commands are passed to memory controller which is able to handle external memory, such as SDR SDRAM or DDR SDRAM. An arbiter is used to arbitrate

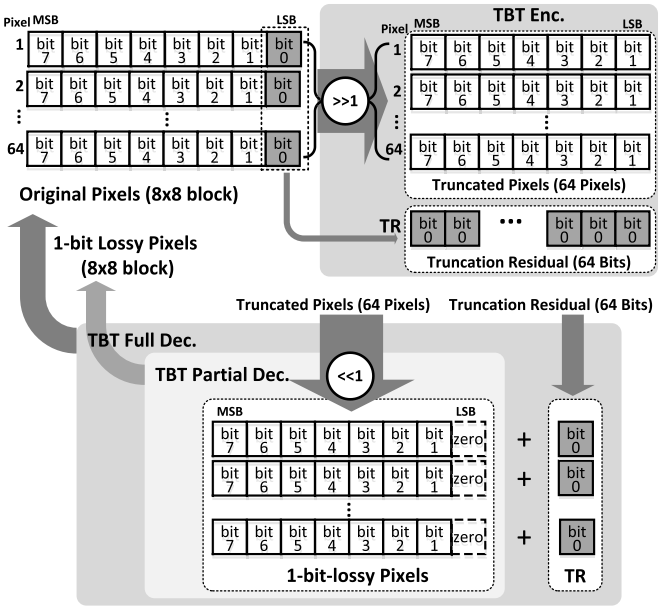


Fig. 4. Tail bit truncation.

different source of memory access. MMU is used to do address mapping from block number to memory address.

For MLL frame recompression part, there are two data paths: the encoding path in right, and the decoding path in left. In encoding path, the filtered pixels are fetched by 8×8 blocks. Three proposed techniques are performed in sequence: TBT, IBP, and SVO-VLC encoding. The outputs of MLL scheme are truncation residual (TR, 64 n bit) and prediction residual (PR, variable length), which will be stored into external memory. In decoding path, the TR and PR are read out separately from external memory. For PR, it is sequentially processed by SVO-VLC decoding, IBP, and left-shifting. The final n-bit lossy 8×8 block is reconstructed. For TR, it is only accessed when lossless pixels are demanded from other modules. The lossless pixels can be reconstructed by adding TR with previously decoded PR.

A. Tail Bit Truncation

Tail Bit Truncation (TBT) is a right-shifting operation to remove the tail bit of original pixels. Take 1-b TBT as an example (Fig. 4), 8×8 original pixels with 8-b sampling are loaded into TBT encoder. For each pixel, the 8-b original data is right shifted 1 b. As a result, two data sets are generated by TBT: truncated pixels and TR. The bit0 of all pixels are assembled together as TR, which will be stored to external memory directly. The remained 7-b data (bit7–bit1) comprise the truncated pixel, which will be output to IBP for further compression. There are two kinds of TBT decoding: partial decoding and full decoding. The truncated 7-b pixels are generated by SVO-VLC decoding and IBP. Partial decoding only left-shift the truncated pixel to 8-b, and add zero to least significant bit (LSB). 1-b lossy pixels can be reconstructed in partial decoding. Full decoding adds one more operation after partial decoding, it replaces the LSB zeros by TR. And the original lossless pixels are reconstructed.

The benefit of TBT is that it almost has no effect to video quality when doing IME under 1-b precision loss of pixel. The pixel value after truncation ranges from 1 to 127. The smaller range of pixel data makes the residual of IBP smaller and produces more zeros than nonzeros, which benefits the compression efficiency of SVO-VLC.

Moreover, TBT can be extended to multibit truncation. For multibit truncation, the compression ratio of truncated pixels is higher than 1-b truncation. However, the video quality will be affected since IME search has lower precision. The detailed discussion of multibit truncation is presented in Section V.

B. Intra Mode Referenced In-Block Prediction

IBP is used to compress the truncated pixels from TBT. To achieve high compression ratio, many prediction methods and modes are used in our scheme. As shown in Fig. 5, the inputs of IBP are 8×8 truncated block. The input pixels are categorized into three types: initial pixel (IP) which located in (0, 0), basic pixels (BP) which located in top and left, and remained are normal pixels (NP). Two main procedures are used in IBP: P1 prediction and P2 prediction. For P1 prediction, it uses IP and BP to do vertical and horizontal DPCM prediction, and generate BP residual. IP is the start point of P1 prediction, as shown in the right part of Fig. 5. The detailed function of P1 is shown as below.

P1 Prediction:

1) Vertical Prediction

$$P_{\text{pred}}(0, x) = BP(0, x-1) \tag{3}$$

$$BP_{\text{Residual}}(0, x) = BP(0, x) - P_{\text{pred}}(0, x). \tag{4}$$

2) Horizontal Prediction

$$P_{\text{pred}}(x, 0) = BP(x-1, 0) \tag{5}$$

$$BP_{\text{Residual}}(x, 0) = BP(x0) - P_{\text{pred}}(x, 0). \tag{6}$$

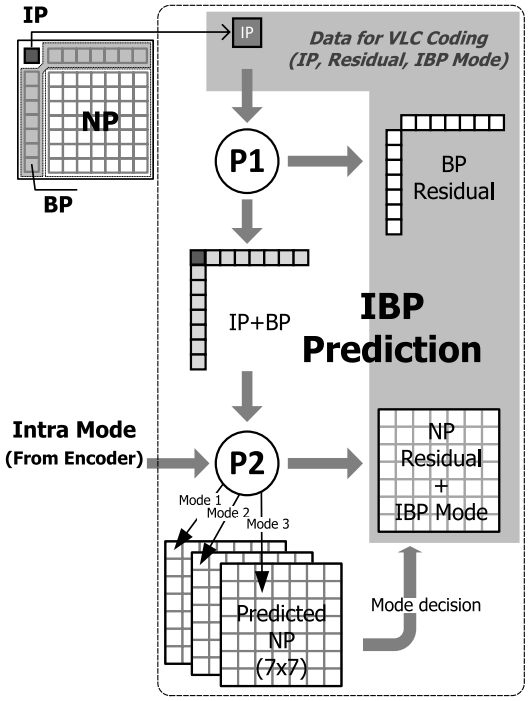
For P2 prediction, it uses both of IP and BP as reference pixels. Intra mode from encoder is used to reduce mode prediction computation. Three candidate modes are selected according to intra mode, and one best NP prediction is chosen after mode decision. As shown in top right part of Fig. 5, every pixel has four reference pixels for prediction. C is the pixel to be predicted, R1 $\sim$ R4 are neighboring reference pixels. The P2 prediction can be described as

P1 Prediction:

$$C_{(x,y)} = f(R1_{(x-1,y)}R2_{(x-1,y-1)}, R3_{(x,y-1)}R4_{(x+1,y-1)}) \tag{7}$$

$$NP_{\text{residual}}(x, y) = NP(x, y) - C_{(x,y)} \tag{8}$$

$f()$ is a prediction function based on IBP mode, and its detailed definition is listed in Table I. To gain a good prediction result, totally eight prediction modes are proposed in this paper. More IPB modes produce more accurate pixels. However, it also brings more mode bits and increases the complexity of mode decision. Considering the massive computation of mode decision, the intra mode from video encoder is



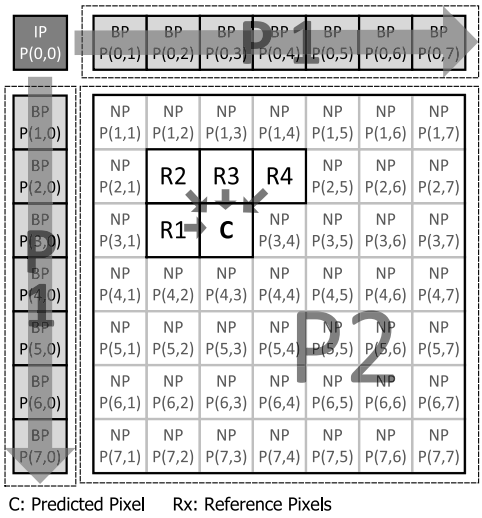


Fig. 5. Intra mode referenced IBP.

TABLE I
IBP MODE

Mode	Prediction (C=)
0	R1
1	R3
2	(R1+R2)/2
3	(R3+R4)/2
4	(R1+R4)/2
5	(R1+R3)/2
6	((R1+R2)/2+R3)/2
7	((R1+R2)/2+(R3+R4)/2)/2

TABLE II
MODE MAPPING FOR HEVC INTRA MODE AND IBP MODE

Intra Mode	IBP Mode	Intra Mode	IBP Mode
Planner	4,5,7	11-18	0,2,6
DC	4,5,7	19-26	1,5,6
2-10	0,4,5	27-34	1,3,4

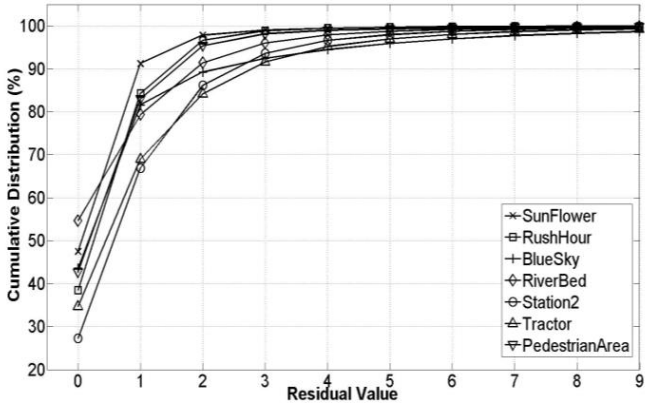


Fig. 6. Cumulative distribution of residuals.

used to do premode decision. Table II shows a mode mapping from HEVC intra modes. There are 35 modes in the HEVC intra prediction. Each intra prediction mode maps to three IBP candidate modes. Because intra prediction is always performed in video encoder, every 8×8 coding block can guarantee an intra prediction mode from encoder. Another effort to reduce IBP complexity is that it only uses addition and shifting, which introduce very little computational overhead in both of CPU and digital circuit.

The output of IBP includes IP, IBP mode, BP residual and NP residual. VLC coding will be performed to reduce size of data. For MLL encoding path, all of operations mentioned above should be performed, whereas for MLL decoding path, it becomes much simpler as it does not needed to do mode decision.

C. Small-Value Optimized Variable Length Coding

Most of the residuals from IBP are small value. As we adopt TBT in MLL scheme, it produces smaller value than other schemes. Smaller value gives the chance to get smaller data after compression. To find a suitable VLC coding method, the residual data is analyzed as shown in Fig. 6. The results come from seven FHD video sequences.

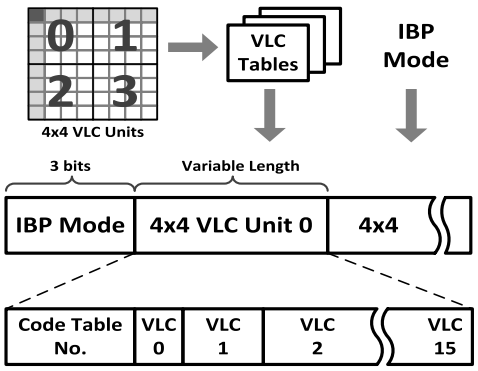


Fig. 7. Small-value optimized variable length coding.

TABLE III
PROPOSED VLC TABLE

	01	**10**	**00**	**110**
Max. value	*0*	*1*	*2*	*3~4*
0	-	1	01	001
±1		0S	1S	01S
±2			00S	10S
±3				11S
±4				000S
	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	xxx…xxS
±12		1100S	11000S	
…		…	…	
±15		1111S	11011S	
±16		00000S	1110000S	
…			…	
±31			1111111S	
			0000000S	

It shows the accumulative absolute date distribution of residual value from 0 to 9. More than 90% residuals are ranging from 0 to 4, and less than 1% residuals exceed 9. Averagely 45% residuals are zero. Another key feature is that small values are highly locational relevance. The neighboring value of a small value is highly attempt to be a small value, while a nonsmall value follows the same rule. Considering these characters, a small-value optimized VLC method is shown in Fig. 7.

To suit the locality of value distribution of residual data, 8×8 residual block is divided into four 4×4 units. Each unit shares one VLC table, and a table index is included in the bit-stream. A special case for unit0 is IP, which is ignored, since IP is a big data in most of cases. There are eight VLC code tables in our scheme, as shown in Table III. If the residual is bigger than 32, the value is encoded by Exp-golomb. *S* indicates the sign of the residual. The coding table is selected according to the maximum absolute value of each 4×4 block. Then the residuals are encoded with the corresponded code table.

The final PR bit stream is packed as shown in Fig. 7. The first part is IBP prediction mode. The followings are four 4×4 VLC units. In each 4×4 VLC unit, the code table index is encoded at first, and then follows 16 VLC data.

D. External Memory Organization

The external memory organization is shown in the right part of Fig. 3. There are two memory banks for reference pixel storage: PR bank and TR bank, and one memory bank for address table (TLB) storage. Two types of addressing method are used in our design: TLB addressing and fixed addressing. For fixed length data, such as TR and TLB data, it can be addressed by their base address and block index. For PR data, the data size is variable. To continuously load and store data, PR data is byte aligned, and the TLB is used to record start address and length of each 8×8 block.

For FHD video sequence, there are totally 32 640 8×8 partitions. It is not a good idea to store all of these TLBs on-chip. A small on-chip TLB SRAM with a separate TLB memory bank is required. Before fetching PR data, the corresponding TLB should be loaded in advance. Even it causes overhead, considering PR data is much more than TLB data, this cost is acceptable. Furthermore, the search window (SW) can be loaded row by row, and it can only preload the leading and ending block's TLB to further reduce overhead.

To improve the external memory efficiency, TLB data is better to be accessed continuously. For example, SW can be loaded row by row, only TLBs in vertical boundary are needed. In this way, the TLB data is stored column by column in TLB memory bank. A typical TLB data for FHD video encoder is provided in Fig. 3.

A simple MMU is used to handle all of memory addressing and TLB refreshing. The 8×8 block entry address for luma and chroma can be generated based on frame number, PR/TR base address, luma/chroma selector, block coordinate, and on-chip TLB SRAM.

IV. EVALUATION OF MLL SCHEME

A. Compression Ratio

To evaluate the compression ratio of the proposed MLL scheme, seven FHD videos with 4:2:0 sampling are applied as test sequences. As a measure of the compression performance, the compression ratio is defined as

$$\text{CR} = \frac{\text{compressed frame size}}{\text{original frame size}}. \tag{9}$$

The CR results of selected sequences are shown in Table IV. For 0-b TBT truncation ($n = 0$), it is lossless compression, and it achieves 45.4% compression ratio, which outperforms current designs by 2.6%–5%. If it uses 1-b TBT, the compression ratio can be improved to 32.4%. The most significant case (Sun flower) achieves 25% compression ratio, which means 3/4 bandwidth can be saved if it allowed 1-bit precision loss.

TBT is a key technology in MLL to improve compression ratio and save bandwidth. It can be extended to multibit. In n-bit ($0 < n \leq 8$) TBT case, then least significant bits of pixel are truncated and assembled as TR. For each 8×8

TABLE IV
COMPRESSION RATIOS AND TR/PR RATIOS UNDER MULTIBIT TBT

Sequence		Sun flower	Blue sky	Station2	River bed	tractor	Pedestrian area	Rush hour	Average
PR Compression Ratio	n=0	0.381616	0.427008	0.484855	0.534561	0.495971	0.416645	0.440156	0.454402
	n=1	0.250115	0.301489	0.344788	0.399781	0.374643	0.294862	0.302265	0.323992
	n=2	0.173653	0.216552	0.222972	0.265164	0.262573	0.188235	0.19134	0.217213
	n=3	0.129750	0.149596	0.157324	0.183777	0.178852	0.139342	0.142568	0.154458
TR/PR block number Ratio	n=1	5.55%	5.75%	5.42%	13.16%	7.15%	8.08%	7.16%	7.47%
	n=2	6.44%	6.13%	5.92%	13.22%	7.46%	8.80%	7.79%	7.97%
	n=3	8.44%	6.60%	9.09%	13.18%	8.63%	9.56%	8.81%	9.19%

block, there are 64 n bits TR. The truncated (8-n) bit pixels are handled by IBP and SVO-VLC. With the increasing of n, the difference of neighboring truncated pixels is decreasing, which is beneficial to the IBP compression. As a result, the compression ratio of PR can be significantly increased. From Table IV, compress ratio under 3-b TBT is as low as to 15.4%, which is inconceivable for traditional compression methods. On the other hand, multibit TBT causes TR growing, which means more TR data is needed for FME and MC.

The principle of multibit TBT selection can be according to IME complexity. For a fast algorithm with few search points, it selects less-bit truncation to improve single pixel precision. While for algorithm with mass search points, it selects more-bit truncation to reduce memory bandwidth. Since IME occupies much more external memory access than FME and MC, most of reference data are fetched under n-bit loss. As a result, the actual compression ratio tends to be very low.

B. Bandwidth Reduction Ratio

The bandwidth cost of encoder mainly comes from two sources: 1) reading reference pixels for motion estimation and motion compensation and 2) writing reconstructed pixels. To evaluate the real bandwidth reduction ratio, it needs to take PR, TR, and TLB overhead in consideration. The overhead includes: byte-alignment (B_o) and TLB. As PR is VLC coded, it should to add extra bit (1b $\sim$ 7b) to make PR byte-aligned. As it follows even distribution, B_o equals to 3.5 b. TLB is used to addressing PR for SW loading. It is possible to only load/store few of PRs TLB to save bandwidth cost, and the details will be discussed in next section. The block amount ratio of TR/PR mainly depends on search window size and the motion activity of video sequence. The statistics of TR/PR ratio from seven FHD video sequences with search range (-128, 127) have been made, as shown in Table IV.

MLL contributes a lot for reading bandwidth reduction (BR_r), because most of TRs can be saved in SW. The reduction ratio is

$$\mathrm{BR}_r = \frac{\mathrm{PR} + B_o + \mathrm{TLB} * r_{\mathrm{tlb}} + \mathrm{TR}_n * r_{\mathrm{pt}}}{64 \times 8} \times 100\% \quad (10)$$

where r_{tlb} is the ratio of TLB should be loaded for each PR in SW, r_{pt} is the block amount ratio of TR over PR, and TR_n is the TR data size under n-bit truncation.

Different from reading, encoder should write all of compressed data (TR + PR) to external memory. The writing bandwidth reduction (BR_w) highly depends on compress ratio

$$\mathrm{BR}_w = \frac{\mathrm{PR} + B_o + \mathrm{TLB} * r_{\mathrm{tlb}} + \mathrm{TR}_n}{64 \times 8} \times 100\%. \quad (11)$$

The final bandwidth reduction ratio ($\mathrm{BR}_{\mathrm{total}}$) is

$$\mathrm{BR}_{\mathrm{total}} = \mathrm{BR}_r \times r_{\mathrm{rd}} + \mathrm{BR}_w(1 - r_{\mathrm{rd}}) \quad (12)$$

where r_{rd} is the reading bandwidth ratio of encoder, r_{rd} is highly related to motion estimation algorithms, SW data-reuse methods, SW size and so on. Normally, the reading bandwidth occupies more than 80% of video encoder.

Table V compares our MLL scheme with other schemes. The compression ratio is the mathematical performance of different algorithms, and the bandwidth reduction ratio is the final IO data reduction. Taking HEVC FHD video encoder with (-128, 127) search range, Level C data reuse as an example, the reading bandwidth reduction ratio is 19.8%, and the writing is 53.8%. Reading occupies about 83% of bandwidth. The final bandwidth reduction is 25.5%.

Bao *et al.* [8] combined DPCM compression and data-reuse cache together to achieve 30.8% bandwidth reduction ratio, whereas the data compression ratio is still about 50%. Cheng *et al.* [20] combined SPIHT and DWT in their method. Tsai and Lee [19] proposed a compression method for display memory; however, it cannot be used in video codec. Our method is equivalent to lossless compression, since the original pixel can be reconstructed by PR (base layer, BL) and TR (enhancement layer, EL). Because only our algorithm has two sets of data after compression, each of them can be separately used in motion estimation and motion compensation. The bandwidth reduction ratio of our design can be much lower than compression ratio. The final result includes data alignment, TLB overheads. As other designs have not fully considered those overheads, our design further outperforms them in practical usage.

MLL only causes IME prediction accuracy degradation, and it does not introduce any loss in FME and MC. In contrast, other lossy compression methods will introduce precision loss in FME and MC, which causes much higher degradation in video quality than ours. From Table V, the PSNR reduction of our design is negligible while n less than 3. According to

TABLE V
COMPARISONS OF DIFFERENT FRAME RECOMPRESSION SCHEMES

	HDM [11]	I-ADPCM [7]	ADPCM [10]	ERP [12]	*Bao* [8]	*Tsai* [19]	*Cheng* [20]	Proposed
Compression Type	Lossless	Lossy	Lossy	Lossy	Lossless	Lossless or Lossy	Lossless or Lossy	Lossless (BL+EL)
Compression Ratio	48%	<50%	48% [16]	50%	50%	60% @ lossless 31%~41%@lossy	50% @lossless 39%, 23%@lossy	45.4% (n=0) ~ 52.9% (n=3)
ME Prediction	Lossless	Lossy	Lossy	Lossy	Lossless	Depends on compression	Depends on compression	Partial lossy
MC Reconstruction								Lossless
Bandwidth Reduction Ratio	48%	<50%	48% [16]	50%	30.80%	60% @ lossless 31%~41%@lossy	50% @lossless 39%, 23%@lossy	25.5% (n=3)
Block Size	8x8	4x4	8x8	2x2	8x8	Frame	16x16	8x8
Target Resolution	1080p	1080p	720p	720p	4K	4K	VGA	1080p
Target Application	Encoder	Encoder	Decoder	Decoder	Encoder	Display MEM	Encoder Decoder	Encoder
Target Standard	H.264 /AVC	H.264 /AVC	MPEG-1	H.264 /AVC	H.264 /AVC	/	/	HEVC
Codec Relativity	/	Intra	/	/	/	/	/	Intra, ME
PSNR Reduction	0	-1.03	-6.58 [5, 7]	-0.472	0	-0.01 ~ -3.13	0 ~ -2.1	-0.01 (n=3)
Year	2009	2007	1998	2008	2010	2010	2009	2013

our experiments, even set $n = 6$, it only causes 0.14 db PSNR reduction.

C. DRAM Bandwidth Utilization Discussion

Bandwidth reduction ratio is a theoretical number to show how much data can be saved from external memory. In the real case, it should concern the bandwidth utilization of external memory, especially for DRAM.

DRAM, such as SDRAM or DDR1/2/3, is widely used as external memory for its high capacity and low price. Unlike SRAM, DRAM prefer block-based transmission, and random access is harmful. Especially for DDR series DRAM, the prefetch operation greatly speedup the data transmission. However, if the desired data is tiny and fractional distributed in external memory, most of the prefetched data are useless which result in very low bandwidth utilization.

For real usage of frame recompression, it should seriously consider to improve the DRAM bandwidth utilization after data compression. The basic principle is to merge the fractional data to big chain, and make the adjacent data continued store in external memory. For video encoder, motion estimation algorithms, data-reuse methods and external memory organizations are key facts for bandwidth utilization.

As shown in Fig. 8, three typical implementations of video encoder with different data-reuse methods are listed: 1) nondata reuse which needs to load whole SW for every current LCU; 2) data-reuse level C which only needs one LCU column in SW; and 3) wave-front parallel processing (WPP) [1] which needs one column and one row of LCU. No data reuse is very flexible for encoder to set search center and SW size. Level C and WPP normally fix search center to (0, 0), otherwise it causes irregular SW overlap and makes memory management difficult.

The gray point shown in Fig. 8 represents a basic processing unit (BPU) in MLL, which is 8×8 pixel array as shown in Fig. 5. A BPU includes two sets of data: PR and TR. LCU is a 64×64 pixel array, which includes 8×8 BPUs. For a $(-128, 127)$ SW, it includes 5×5 LCUs which equals to 40×40 BPUs. The acquired pixels from external memory are highly depends on data-reuse methods. Only gray points are need to be loaded from external memory. To merge as many BPUs together as possible, three memory organization methods are proposed: ROW-based, COLUMN-based, and LCU-based.

As shown in the figure, ROW/COLUMN-based means one row/column of BPUs in a frame are continuously stored in external memory. The corresponding TLB for each BPU are inversely (column/row) stored. The (x, y) represents the coordinate of BPU. LCU-based is 8×8 BPUs in one LCU, leading and ending TLBs are continuously stored in external memory. ROW- and COLUMN-based organizations support very flexible search center and SW size. LCU-based organization will cause overhead if the boundaries of SW and LCU do not match.

Memory organization helps to merge BPUs as a long chain to improve the bandwidth utilization of DRAM. Table VI shows the chain length of BPU and TLB under different data-reuse methods and memory organizations. (BPU, TLB) shows how many BPUs and TLBs can be loaded/stored at a time. From our experiments, for each BPU, PR is about 79 b (Table IV), TR is fixed to 192 b ($n = 3$), and TLB is fixed to 24 b (Fig. 3).

As discussed in Section IV-B, the main data access comes from PR. It should set the chain length of PR (or BPU) as long as possible. Taking Level C data reuse in Fig. 8 as an example, Column-based or LCU-based memory organization produces

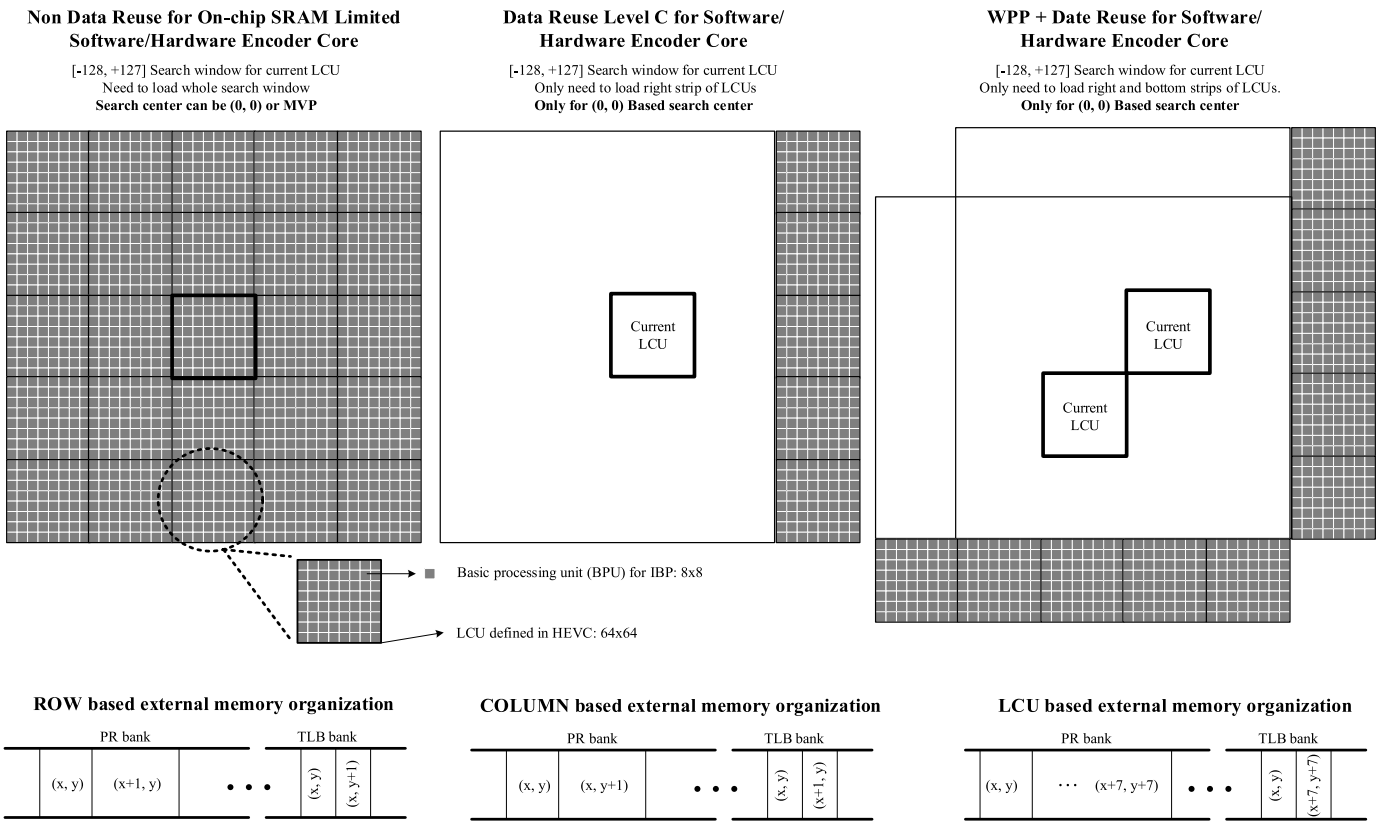


Fig. 8. Data reuse and memory organization for improving DRAM bandwidth utilization.

TABLE VI
CHAIN LENGTH OF (BPU, TLB) UNDER DIFFERENT DATA-REUSE METHODS AND EXTERNAL MEMORY ORGANIZATIONS

Memory organization	Non Reuse	Level C Reuse	WPP+Level D Reuse	Fractional LCU RW support
Row Based	(40, 40)	(8, 40)	(8,40) or (40,8)	YES
Column Based	(40, 40)	(40, 8)	(8,40) or (40,8)	YES
LCU Based	(64, 10)	(64, 10)	(64, 10/2)	NO
Search center support	(0,0) or MVP	(0, 0)	(0, 0)	/

40 or 64 BPU chain length, which means the average data size of PR is 3160 or 5056 b for one-time DRAM access. The corresponding TLB access is 192 or 240 b. For TR, it depends on the final motion vectors from IME. According to our experiments, less than 8% of TR is pure isolated from neighbors. Most of TRs can be merged into big block data. On the other hand, even if TR is overloaded, it can be reused at the next-time motion estimation. Since the total SW loading is about 27.8 kb (under Level C reuse), TLB and TR only occupy about 10% of total bandwidth. The final DRAM bandwidth utilization still be very high. Many techniques can further improve TR and TLB bandwidth utilization, such as overloading for data reuse between current LCU, data buffering to merge tiny data into big block for writing.

D. IBP Mode and Mode Mapping

DPCM-based frame recompression methods only use two scan patterns (vertical and horizontal), which means they only use two direction information. In contrast, our proposed IBP has seven directional modes and one dc mode as shown in Fig. 9 and Table I. It greatly improves the precision of prediction, and also outperforms DPCM-based compressions.

MLL make uses of intra mode from encoder. As encoder always do intra prediction in both of I and P/B frames, there is no extra overhead to send intra mode information to MLL for encoder. By referring intra mode, 62.5% mode decision computation cost can be saved, as it only needs to compare three out of eight mode in current MLL scheme.

HEVC has very wide intra modes, which provides good chance to find a good mapping between intra mode and IBP mode. Through statistical analysis from seven FHD video sequences, we get the empirical mapping table in Fig. 5. According to our experiments, more than 95% of min-cost mode can be successfully covered if three candidates are selected.

E. Computation Analysis

The main computation in MLL is IBP, which is an extremely light prediction method for both of software and hardware

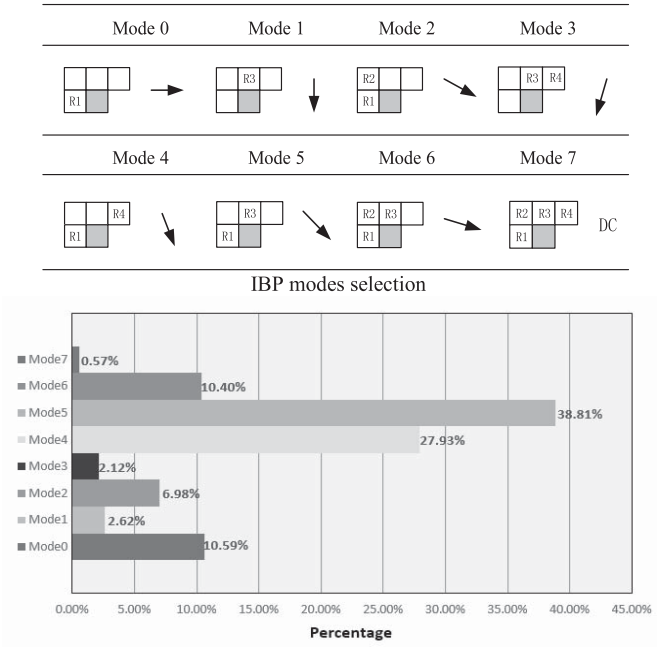


Fig. 9. IBP mode selection and distribution.

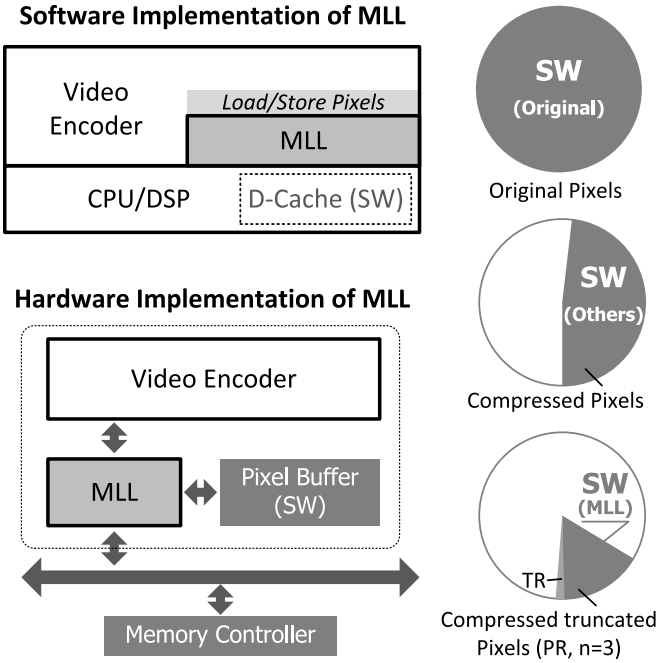


Fig. 10. Implementations of MLL scheme with video encoder.

implementation. Only addition, shifting, and subtraction are needed, and the computation of different modes is different. Fig. 9 shows the average mode distribution under seven FHD test sequences. The average computation of IBP encoding for one 8×8 block is about 334 times $(8 - n)$ bit addition and 144 times right-shifting. The computation of decoding is one-third of encoding.

The execution time of MLL encoding depends on the parallelism of IBP predictor. For three-mode parallel design, it only costs 72 clock cycles to encoding an 8×8 block whereas it is tripled on choosing one-mode serial design. For decoding, it only costs 56 clock cycles. Multiple IBP encoder/decoder engines can work together to improve data throughput.

V. Implementations

A. Software and Hardware Implementation

As shown in Fig. 10, for software implementation, MLL can be implemented as a pixel load/store stack to manage all of reference pixel read/write operations. For hardware implementation, MLL is an interface of video encoder to connect with system bus. All of pixel addressing, compression are done by MLL, and encoder just treats it as a memory.

To evaluate hardware cost and performance, our design is implemented in SMIC 130nm CMOS technology. One IBP encoder engine and two IBP decoder engines are used to balance writing and reading bandwidth. The total hardware cost is 110.4 K Gates, and the maximum clock frequency achieves 250 MHz. Some extra on-chip memory are needed, such as TLB SRAM, VLC table, and SRAM buffer for temporary VLC data. They totally cost about 5 KB. The external memory size can be reduced to 60% of original frame size for worst case consideration. To support real-time FHD at 30 frames/s, (−128, 127) search range encoding, the minimum working frequency for IBP decoding is 206, and 106 MHz for IBP encoding.

TABLE VII
Average Data Transmission Length and Bandwidth Requirement Ratio

	Average data length		Bandwidth requirement Ratio
	Read	Write	
PR	5Kb	5Kb	90%
TR	0.5Kb	1.5Kb	9%
TLB	240b	48b	1%

As discussed in Section IV-C, DRAM utilization is a key issue to evaluate the usability of frame recompression. There are a lot of DRAM chips that can provide various bandwidth performance, such as SDRAM, DDR 1/2/3. The external memory IO bit-width also can be extended by using multiple DRAM chips together to provide multiplied bandwidth. To evaluate the DRAM utilization, the bandwidth requirement should be estimated first. For HEVC FHD at 30 frames/s, (−128, 127) search range and level C data-reuse encoder, the total reading bandwidth requirement is about 500 MB/s, and writing is about 100 MB/s. After MLL compression, the total bandwidth can be reduced to less than 150 MB/s. If it uses a dedicated memory, 16 b DDR2 or 8 b DDR3 is more than enough. Moreover, since bandwidth requirement has been greatly reduced, SDRAM or DDR1 also can be a choice for FHD video encoder.

RTL simulation is performed under FHD at 30 frames/s, (−128, 127) search range, LCU-based memory organization

TABLE VIII
AVERAGE PSNR AND BITRATE DEGRADATION UNDER MULTIBIT MLL

Algorithm	n-bit TBT	ΔPSNR (db)				ΔBitrate (%)			
		QP=22	QP=27	QP=32	QP=37	QP=22	QP=27	QP=32	QP=37
Full	1	0.00055	-0.00098	-0.00327	-0.00087	0.009696	-0.03833	0.089302	-0.08428
	2	-0.00305	-0.00417	-0.00702	-0.00866	0.097268	0.076796	0.006023	0.035364
	3	-0.00669	-0.01018	-0.01614	-0.01742	0.595862	0.362562	0.230156	0.058681
	6	-0.06796	-0.14792	-0.23736	-0.27708	10.13551	11.1151	10.77613	9.788198
4SS	1	-0.00072	0.00224	0.001093	-0.00699	-0.08447	0.14133	0.136354	0.030642
	2	0.000336	-0.00026	-0.00233	-0.00703	0.139997	0.191418	0.049622	-0.02838
	3	-0.00292	-0.00327	-0.01089	-0.01599	0.547941	0.531535	0.170231	0.229826
	6	-0.03273	-0.07552	-0.12018	-0.1543	6.268275	6.727117	6.116871	5.60645
TZ	1	-0.00104	-0.00243	-0.0033	-0.0038	-0.03462	-0.0893	0.071328	-0.15111
	2	-0.00166	-0.00454	-0.00362	-0.00376	0.06602	0.033003	0.138384	-0.03081
	3	-0.00451	-0.00598	-0.01108	-0.01068	0.495276	0.277164	0.097277	-0.0608
	6	-0.05961	-0.11802	-0.17425	-0.21534	7.416347	7.915263	7.272342	6.629371

and Level C data reuse. SAMSUNG K4B1G0846F-HCF8 DDR3 RTL Model, Altera UNIPHY DDR3 controller [21], AXI [22] 64 b data bus are used for test environment. Three IO bit-width configurations, 8, 16, and 32 b, are tested. Table VII shows the average data transmission length and bandwidth distribution. The average bandwidth utilization, which is calculated by valid data over all prefetched data, is 95.5% for 32 b DDR3; 97.7% and 98.7% for 16 and 8 b DDR3 memories, respectively. LCU-based memory organization is the most high efficient for bandwidth utilization. The downside is that it should set search center to (0, 0) and SW boundary should match the LCU boundary.

B. Data-Reuse Strategy With MLL

Frame recompression is not the only way to save bandwidth. Data reuse also contributes a lot for bandwidth saving. To finely save external memory access, the combination of data reuse and frame recompression is necessary for next-generation video coding system. Fortunately, MLL is an inherent method which benefit data reuse.

Data reuse is an efficient method to reuse SW for neighboring LCU (or MB). For each current LCU, the whole search window will be loaded in advance. The overlapped region of SW can be reused for next SW. The *D*-cache and the pixel buffer in Fig. 10 can be used to store SW on chip. Normally, all of original pixels are stored in SW. For conventional frame recompression schemes, the compressed pixels are loaded into SW. For MLL scheme, only PR is stored in SW, and TR will be loaded according to motion vectors (MV). As shown in right part of Fig. 10, by combining with data-reuse scheme, our proposed MLL scheme can reduce SW SRAM size and also bandwidth to lower level.

C. Pixel Truncation Strategy With MLL

Pixel truncation is a well-known method to reduce bandwidth for motion estimation, and it also contributes to save hardware cost since the data precision is truncated. According to [13], IME occupied about 33% of hardware resource in encoder. Taking well-known 2-D SAD-Tree IME architecture [14], [18] as an example, it saves about 35% of hardware cost under 3-b pixel truncation. In the other hand, pixel truncation itself is not a good solution for bandwidth reduction. Three-bit truncation only saves 37.5% bandwidth, and higher bit truncation will causes serious video quality degradation.

MLL can work perfectly with pixel truncation, and it already include it as a part. MLL only introduces precision loss in IME, which keeps lower quality loss than ever. Truncation along with MLL compression greatly reduces redundancy between neighboring pixels, and saves more bandwidth than ever, which is over two times of conventional truncation based designs.

D. MLL-Based ME Versus Conventional ME

To evaluate the impact of MLL-based ME on video quality, the MLL frame recompression scheme is employed in an HEVC encoder. The lossy PR data is used in the IME, and the lossless PR+TR are used in FME and MC. With HM 10.0 reference software [15], three IME search algorithms are employed in this experiment: Full Search (FS), TZ search and four-step search (4SS). FS and TZ search are provided in HM. 4SS is implemented by ourselves. The experiments are performed under the following settings: sequence structure is IBBB (the P frame is also called B frame in HM 10.0), search window is set to $(-64, 63)$, and the resolution of test sequences is FHD. Seven sequences are tested under three IME search algorithms and four MLL compression settings: 1–3 b lossy, and 6-b lossy, as shown in Table VIII. For 1–3 b MLL compression, the video quality is almost the same as lossless one. As far as increasing the truncation to 6-b, the visible quality drop then appears. Chen *et al.* [18] have mentioned that 3-b truncation is almost lossless for IME search in H.264/AVC, and here our experiments confirmed it for HEVC encoding.

VI. Conclusion

In this paper, an encoder friendly frame recompression scheme, MLL, is proposed. This scheme fully uses the information from encoder and discriminates IME from FME and MC to finely save external memory bandwidth. There are two coding layers in our scheme: BL and EL. BL is lossy data only used for IME, and BL + EL are lossless data which is used for FME and MC. The BL data is encoded by IBP and SVO-VLC and the EL data is stored directly in the external memory. Experimental results show that the bandwidth of the external memory can be reduced by 74.5%. The proposed MLL can finely cooperate with video encoder, data reuse methods, and fast search algorithms. It is easy to be integrated into both of hardware and software video encoder.

References

[1] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, *High Efficiency Video Coding (HEVC) Text Specification Draft 10*, document Rec. JCTVC-L1003, 2013.

[2] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[3] *JVT of ISO/IEC MPEG, ITU-T VCEG, MVC Software Reference Manual-JMVC 8.2*, document Rec. JVT-B118r2, May 2010.

[4] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[5] T. Y. Lee, "A new frame-recompression algorithm and its hardware design for MPEG-2 video decoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 529–534, Jun. 2003.

[6] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 4, pp. 461–474, Apr. 2001.

[7] Y. Lee, C.-E. Rhee, and H.-J. Lee, "A new frame recompression algorithm integrated with H.264 video compression," in *Proc. IEEE Int. Symp. Circuits Syst., ISCAS*, May 2007, pp. 1621–1624.

[8] X. Bao, D. Zhou, and S. Goto, "A lossless frame recompression scheme for reducing DRAM power in video encoding," in *Proc. IEEE Int. Symp. Circuits Syst., ISCAS*, May/Jun. 2010, pp. 677–680.

[9] 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 Trans. Multimedia*, vol. 14, no. 2, pp. 237–249, Apr. 2012.

[10] D. Pau and R. Sannino, "MPEG-2 decoding with a reduced RAM requisite by ADPCM recompression before storing MPEG-2 decompressed data," U.S. Patent 5838597, Nov. 17, 1998.

[11] S.-H. Lee, M.-K. Chung, S.-M. Park, and C.-M. Kyung, "Lossless frame memory recompression for video codec preserving random accessibility of coding unit," *IEEE Trans. Consum. Electron.*, vol. 55, no. 4, pp. 2105–2113, Nov. 2009.

[12] Y. V. Ivanov and D. Moloney, "Reference frame compression using embedded reconstruction patterns for H.264/AVC decoder," in *Proc. 3rd Int. Conf. Digit. Telecommun., ICDT*, Jun./Jul. 2008, pp. 168–173.

[13] T.-C. Chen, C.-J. Lian, and L.-G. Chen, "Hardware architecture design of an H.264/AVC video codec," in *Proc. Asia South Pacific Conf. Des. Autom., ASP-DAC*, Jan. 2006.

[14] Y. Fan, X. Zeng, and S. Goto, "Optimized 2-D SAD tree architecture of integer motion estimation for H.264/AVC," *IEICE Trans. Electron.*, vol. E94-C, no. 4, pp. 411–418, Apr. 2011.

[15] *HM 10.0 reference software* [Online]. Available: http://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/trunk/

[16] T. Y. Lee, "A new algorithm and its implementation for frame recompression," *IEEE Trans. Consum. Electron.*, vol. 47, no. 4, pp. 849–854, Nov. 2001.

[17] Z.-L. He, C.-Y. Tsui, K.-K. Chan, and M. L. Liou, "Low-power VLSI design for motion estimation using adaptive pixel truncation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 5, pp. 669–678, Aug. 2000.

[18] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 3, pp. 578–593, Mar. 2006.

[19] T.-H. Tsai and Y.-H. Lee, "A 6.4 Gbit/s embedded compression codec for memory-efficient applications on advanced-HD specification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 10, pp. 1277–1291, Oct. 2010.

[20] C.-C. Cheng, P.-C. Tseng, and L.-G. Chen, "Multimode embedded compression codec engine for power-aware video coding system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 141–150, Feb. 2009.

[21] *Altera UNIPHY DDR3 Memory Controller*. [Online]. Available: http://www.altera.com.cn/products/ip/iup/memory/m-alt-high-perf-mem-controller-ii.html, accessed Feb. 27, 2014.

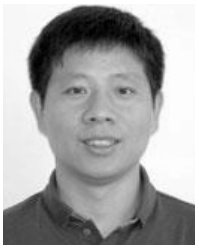
[22] *AXI Bus*. [Online]. Available: http://www.arm.com/zh/products/system-ip/amba/amba-open-specifications.php, accessed Feb. 27, 2014.

Yibo Fan (M'13) received the B.E. degree in electronics and engineering from Zhejiang University, Hangzhou, China, in 2003; the M.S. degree in microelectronics from Fudan University, Shanghai, China, in 2006; and the Ph.D. degree in engineering from Waseda University, Tokyo, Japan, in 2009.

He was an Assistant Professor with Shanghai Jiao Tong University, Shanghai, from 2009 to 2010, and is currently an Assistant Professor with the College of Microelectronics, Fudan University. His research interests include image processing, video coding, and associated VLSI architecture.

Qing Shang received the B.S. and M.S. degrees in microelectronics and solid electronics from Fudan University, Shanghai, China, in 2011 and 2014, respectively.

His research interests include VSLI design, algorithms and VLSI architectures for multimedia signal processing, and designing for tests.



Xiaoyang Zeng (M'05) received the B.S. degree from Xiangtan University, Xiangtan, China, in 1992 and the Ph.D. degree from Changchun Institute of Optics, Fine Mechanics, and Physics, Chinese Academy of Sciences, Changchun, China, in 2001.

He was a Post-Doctoral Researcher with Fudan University, Shanghai, China, from 2001 to 2003. He then joined State Key Laboratory of ASIC and System, Fudan University, as an Associate Professor, where he is currently a Full Professor and the Director. His research interests include information security chip design, system-on-chip platforms, and VLSI implementation of digital signal processing and communication systems.