Lossless Frame Memory Compression Using Pixel-Grain Prediction and Dynamic Order Entropy Coding

Article *in* IEEE Transactions on Circuits and Systems for Video Technology · January 2015
DOI: 10.1109/TCSVT.2015.2469572

4 authors, including:

Zhenyu Liu
Tsinghua University
91 PUBLICATIONS **610** CITATIONS

Lossless Frame Memory Compression Using Pixel-Grain Prediction and Dynamic Order Entropy Coding

Xiaocong Lian, *Student Member, IEEE*, Zhenyu Liu, *Member, IEEE*, Wei Zhou, *Member, IEEE*, and Zhemin Duan

***Abstract*— Power constraints constitute a critical design issue for the portable video codec system, in which the external dynamic random access memory (DRAM) accounts for more than half of the overall system power requirements. With the ultrahigh-definition video specifications, the power consumed by accessing reference frames in the external DRAM has become the bottleneck for the portable video encoding system design. To relieve the dynamic power stresses introduced by the DRAM, a lossless compression algorithm is devised to reduce the external traffic and the memory requirements of reference frames. First, pixel-granularity directional prediction is adopted to decrease the prediction residual energy by 54.1% over the previous horizontal prediction. Second, the dynamic *k*th-order unary/Exp-Golomb rice coding is applied to accommodate the large-valued prediction residues. With the aforementioned techniques, an average data traffic reduction of 68.5% for the off-chip reference frames is obtained, which consequently reduces the dynamic power requirements of the DRAM by 42.3%. Based on the high data reduction ratio of the proposed compression algorithm, a partition group table-based storage space reduction scheme is provided to improve the utilization of row buffers in the DRAM. Consequently, an additional 14.5% of the DRAM dynamic power can be saved by reducing the number of row buffer activations. In total, a 56.8% decrease in the dynamic power requirements of the external reference frame access can be obtained using our strategies. With TSMC 65-nm CMOS logic technology, our algorithm was implemented in a parallel VLSI architecture based on a compressor and decompressor at the cost of 36.5k and 34.7k, respectively, in terms of gate count. The throughputs of the proposed compressor and decompressor are 1.54 and 0.78 Gpixels/s, which are suitable for quad full high definition (4K) @ 94 frames/s real-time encoding with the level-D reference data reuse scheme.**

Manuscript received October 29, 2014; revised March 14, 2015, April 27, 2015, July 8, 2015, and July 23, 2015; accepted August 5, 2015. Date of publication August 18, 2015; date of current version January 6, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 60902101 and in part by the Fundamental Research Funds through the Central Universities under Grant 3102014JCQ01057. This paper was recommended by Associate Editor V. Sze. *(Corresponding author: Wei Zhou.)*

X. Lian, W. Zhou, and Z. Duan are with the School of Electronics and Information, Northwestern Polytechnical University, X'ian 710129, China (e-mail: lianxiaocong123@mail.nwpu.edu.cn; zhouwei@nwpu.edu.cn; zhemind@nwpu.edu.cn).

Z. Liu is with Tsinghua National Laboratory for Information Science and Technology, Research Institute of Information Technology, Tsinghua University, Beijing 100084, China (e-mail: liuzhenyu73@mail.tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2469572

***Index Terms*— Frame memory compression, High Efficiency Video Coding (HEVC), lossless compression, low power.**

I. Introduction

SINCE the draft of H.264/Advanced Video Coding (AVC) [1] was released in 2003, people have witnessed the explosive growth of video in workplaces and entertainments [2]. To meet the high-quality requirements of human perception, video resolution has dramatically increased. Standard-definition and high-definition (720p HD) broadcasts are being replaced by Full HD (1080p), while quad full HD (4K) and super hi-vision (8K) applications are beginning to increase in popularity [3], [4]. These factors have led to the development of the video-coding standards toward higher compression efficiency to support HD and ultrahigh definition (UHD) videos. In this context, High Efficiency Video Coding (HEVC) was developed as an advanced successor of H.264/AVC [5]. The major goal of HEVC is to achieve higher coding efficiency compared with previous standards, especially when operating on high-resolution specifications. The first version of HEVC was ratified in January 2013, and its main objective is to double the compression efficiency at the cost of a complexity increase of 2–4 times compared with H.264/AVC.

Considering the high image resolution, the off-chip dynamic random access memory (DRAM) is applied as the frame memory to obtain the best system cost/performance tradeoff [6]. The obstacles of off-chip reference frame storage originate from two aspects, i.e., bandwidth and power limitations. The bandwidth limitation can be ameliorated by an efficient search region data reuse scheme. For example, 32-bit DDR3-1600 can meet the bandwidth requirements of one-reference-frame motion estimation for videos with 8K @ 60 frames/s specifications by applying the level-D search window [7]. For mobile applications, the power requirements of DRAM are a more serious bottleneck compared with the bandwidth limitation. From the analysis in [8], more than 20% of the system power, which is almost equal to the power of the video codec engine, is consumed by DRAM in smartphone devices [9]. The dynamic power of DRAM consists of three main components:

1) the power required to activate the row buffer (P_{ACT});
2) the internal power consumed by data transitions between the row buffer and IO drivers (P_{RW});
3) the power of IO terminal drivers (P_{IO}) [10].

The reference frame applies the block-based storage. Compressing the data in every data block contributes to reductions in not only the bandwidth requirements but also the signal transitions, and subsequently reduces the power requirements of P_{RW} and P_{IO}. In addition, if the reference frame storage space is compressed, the utilization of the row buffer can be improved; the power requirements of P_{ACT} decrease accordingly.

To reduce the memory bandwidth, many memory compression algorithms have been proposed [11]–[25]. Previous methods can be divided into two categories: 1) lossy compressions and 2) lossless compressions. In general, lossy algorithms [11]–[17] can achieve higher data reduction ratios (DRRs) than lossless algorithms by sacrificing video-coding performance. For example, using the level-C data reuse scheme, trailing-bit truncation, encoder-oriented in-block prediction, and small value optimized variable length coding, the mixed lossy and lossless (MLL) algorithm [17] can reduce the external bandwidth by 74.5% at a cost of 0.01 dB coding quality loss. In addition, the lossy compression can reduce memory requirements. For instance, Lee *et al.* [12] introduced adaptive quantization to guarantee a no less than 50% DRR. Consequently, the memory requirements were reduced by 50%, and the picture quality was degraded by 0.11–1.78 dB. It can be observed that the main drawback of the lossy compression algorithm originates from the drifting problem. Specifically, the mismatch of the reference pictures between the decoder and the encoder, which is generated by the lossy compression, will introduce errors during the prediction procedure, and these errors will be accumulated in consecutive frames. Therefore, the reconstructed video quality is greatly deteriorated.

Lossless algorithms [18]–[26] can circumvent the aforementioned drifting problem. Joint Photographic Experts Group C Lossless Standard [26] is a widely used standard for the lossless and near-lossless compression of images, achieving up to 70.9% DRR. However, the intensive computational complexity and, especially, the high processing latency hinder the standard algorithms as the frame memory compressor. Many algorithms have been proposed to achieve a good balance among the computational complexity, the processing latency and the compression performance [21]–[23]. For example, a lossless embedded compression achieving a DRR of 57.3% was proposed in [22] and consists of a hierarchical prediction method based on pixel averaging and pixel copying and a significant bit truncation (SBT). Multimode differential pulse code modulation (DPCM) and averaging prediction and semifixed length (MDA and SFL) entropy coding were proposed in [27], of which the averaged achieved DRR was as high as 61.9%.

Although previous lossless algorithms can achieve an average of 24%–61.9% DRR, they have the following disadvantages.

1) Because of the variable DRR, the previous works merely decrease the off-chip data traffics but cannot contribute to reducing memory requirements.
2) The VLC algorithms in previous works mainly focused on the compression performance for small values. For higher bit depths, for example, 10 or 12 bits/sample, the amplitudes of the residues increase from the inherent noises. Accordingly, the performances of previous entropy coding methods are seriously degraded.
3) To increase accuracy, the prediction algorithm in [17] adopted auxiliary information from the intra-coding component in the encoder. However, to speed up the encoding procedure, not all 8×8 CUs undergo the intra-mode coding exploration [28], [29]. From the decoder aspect, the auxiliary information is unavailable if the 8×8 Coding Unit (CU) is not coded in intra mode. Therefore, such a prediction method cannot be seamlessly integrated with any encoder, especially for the decoder.

To overcome the above obstacles, we first develop a pixel-grain directional prediction method to further reduce the power of the prediction residuals. Next, we adopt a dynamic *k*th-order unary/Exp-Golomb rice coding to improve the compression rate when addressing large residue values. Finally, we propose a partition group table (PGT)-based storage scheme. The contributions of this paper are summarized as follows.

1) A self-contained directional intra-prediction algorithm is provided. This algorithm possesses two primary features: first, the energy of the prediction residues is reduced by 54.1% compared with the original vertical/horizontal prediction. Second, the prediction angle is deduced using the texture correlations, which discards the dependency on the encoder or decoder auxiliary information to improve the applicability.
2) Dynamic *k*th-order unary/Exp-Golomb rice coding is adopted to achieve comprehensive adaptability, especially when processing large-valued prediction residues in high-bit-depth videos, such as the 30 bpp sequences *NebutaFestival* and *SteamLocomotiveTrain*.
3) Based on the high compression rate of the proposed compression algorithm, the PGT-based storage scheme is used to reduce not only the memory space requirements but also the dynamic power requirements of the DRAM.
4) Finally, the corresponding VLSI architectures are developed to implement the compressor and the decompressor. In the prediction engine design, the algorithm and circuit co-optimization reduces the hardware costs by 61% while improving the speed by 11%. For the compressor VLSI design, the parallel architecture is introduced to enhance the throughput. In the decompressor design, the luminance and chrominance alternate processing can effectively avoid the bubbles in pipelining.

The remainder of this paper is organized as follows. In Section II, we introduce the 16×16 partition-based compression algorithm. The PGT-based storage scheme is explained in Section III. The related hardware implementations of the compression and decompression modules are described in Section IV. Section V illustrates the experimental results. Finally, the conclusion is given in Section VI.

II. 16×16 Partition Compression Algorithm

Thoroughly considering the data dependency induced by deblocking filtering and the DRAM burst access properties,

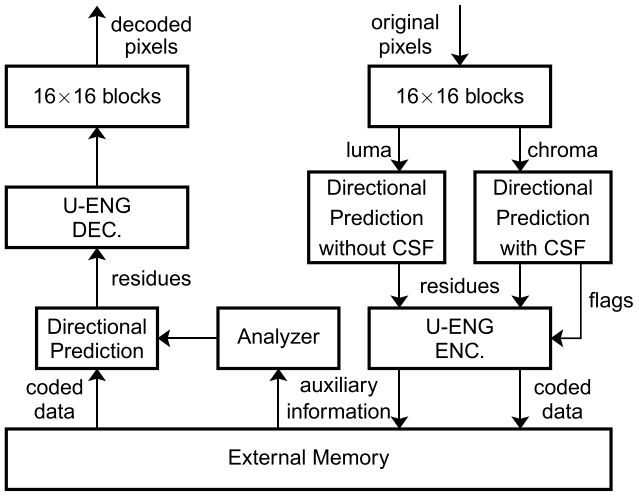

Fig. 1. Block diagram of the compression and decompression algorithms.

we define the basic luma partition size as 16 × 16 (16-pixel column by 16-pixel row), and the corresponding chroma partition as 8 × 8 with a 4:2:0 sampling format.

Fig. 1 shows the block diagram of the compression and decompression algorithms. The improvement of the compression rate originates from the signal prediction accuracy and entropy coding efficiency. Section II-A describes the proposed directional prediction method. Sections II-B and II-C explain the dynamic kth-order unary/Exp-Golomb rice coding algorithm and the significant flag in the chroma block, respectively. Finally, the algorithm optimizations for VLSI implementation are provided in Section II-D.

A. Self-Contained Directional Intra Prediction

Intra prediction is a conventional method used in memory compression algorithms that can reduce the entropy of the coded symbols. The studies in [12], [27], and [30] all applied such an approach. However, the previous works directly calculated the prediction signals from the vertical or horizontal neighboring pixels, which forfeits the compression performance with the low prediction accuracy. The study in [17] proposed eight types of 8 × 8-block-based intra-predictions modes, and the candidate type was determined using the rate distortion optimization result of the HEVC intra encoder. Although the method can increase the prediction efficiency, the reference information can only be obtained from CUs, which has been tested with intra-coding modes. Hereafter, it is difficult to apply this algorithm to compress the CUs, which skip the intra-coding exploration. Especially for the decoder side, for the P and B mode CUs, the required auxiliary prediction information is not available.

To overcome the above hindrances, this paper proposes a self-contained directional intra prediction to obtain increased accuracy [31]. As shown in Fig. 2, the pixels in the black rectilinear block all apply the directional prediction. During the decompression stage, the pixels on the right and bottom boundaries of the current pixel ($p_{i,j}$) are unavailable. Therefore, we estimate the edge direction of the current pixel

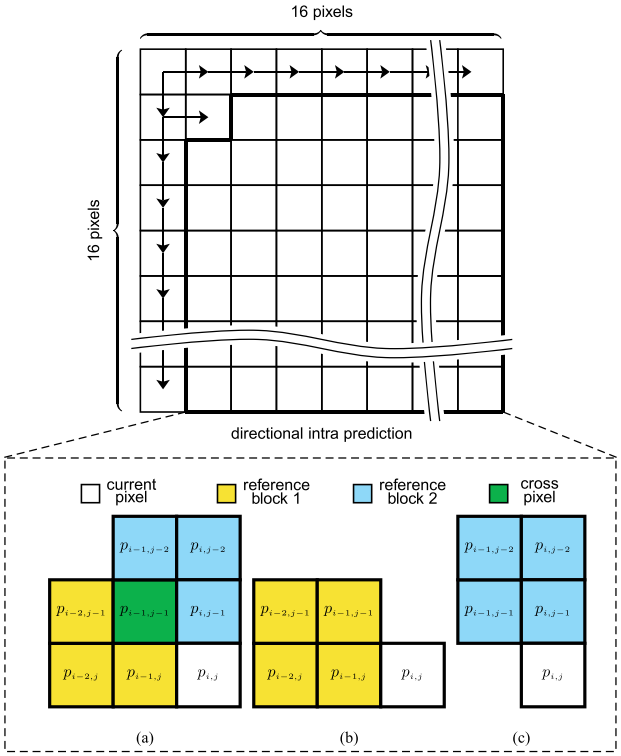

Fig. 2. Pixel locations in one 16 × 16 partition and corresponding prediction modes with (a) both reference blocks, (b) only the left reference block, and (c) merely top reference block.

from its left and top 2 × 2 blocks, as shown in Fig. 2(a). For the current pixel $p_{i,j}$, we define the corresponding left and top neighboring edge vectors as $\vec{D}^l{}_{i,j} = \{dx^l{}_{i,j}, dy^l{}_{i,j}\}$ and $\vec{D}^t{}_{i,j} = \{dx^t{}_{i,j}, dy^t{}_{i,j}\}$, respectively, which are derived from the neighboring pixels as

$$dx^l_{i,j} = p_{i-2,j} + p_{i-1,j} - p_{i-2,j-1} - p_{i-1,j-1}$$
$$dy^l_{i,j} = p_{i-1,j-1} + p_{i-1,j} - p_{i-2,j-1} - p_{i-2,j} \quad (1a)$$
$$dx^t_{i,j} = p_{i-1,j-1} + p_{i,j-1} - p_{i-1,j-2} - p_{i,j-2}$$
$$dy^t_{i,j} = p_{i,j-2} + p_{i,j-1} - p_{i-1,j-2} - p_{i-1,j-1}. \quad (1b)$$

Therefore, the strength of the edge vectors can be roughly estimated by

$$\mathrm{Amp}(\vec{D}^l_{i,j}) = |dx^l_{i,j}| + |dy^l_{i,j}|$$
$$\mathrm{Amp}(\vec{D}^t_{i,j}) = |dx^t_{i,j}| + |dy^t_{i,j}|. \quad (2)$$

The block with the larger strength value is used to define the prediction angle. Specifically, the corresponding prediction angle $\vec{D}_{i,j} = \{dx_{i,j}, dy_{i,j}\}$ is defined as follows:

$$\vec{D}_{i,j} = \left(\mathrm{Amp}(\vec{D}^l_{i,j}) > \mathrm{Amp}(\vec{D}^t_{i,j})\right) ? \vec{D}^l_{i,j} : \vec{D}^t_{i,j} \quad (3)$$

in which $dx_{i,j}$ and $dy_{i,j}$ represent the edge strength in the vertical and horizontal directions, respectively. From $\vec{D}_{i,j}$, the edge direction of the current pixel can be estimated using the ratio between $dy_{i,j}$ and $dx_{i,j}$. We first define the variable $\eta(\vec{D}_{i,j})$ as

$$\eta(\vec{D}_{i,j}) = \frac{dy_{i,j}}{dx_{i,j}}. \quad (4)$$

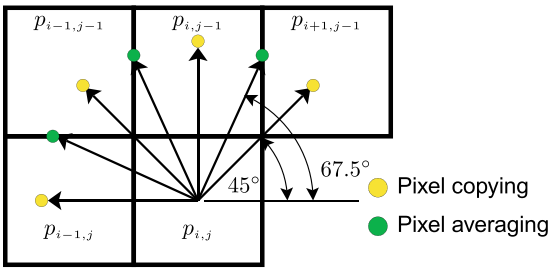

Fig. 3. Seven calculated directions of the residuals.

Then, the value of η is divided into seven bands, as shown in Fig. 3, and the corresponding edge directions are derived as

$$\theta = \begin{cases} 45°, & \text{if } 0.414 < \eta(\vec{D}_{i,j}) \le 1.500 \\ 67.5°, & \text{if } 1.500 < \eta(\vec{D}_{i,j}) \le 5.027 \\ 90°, & \text{if } |\eta(\vec{D}_{i,j})| > 5.027 \\ 112.5°, & \text{if } -5.027 \le \eta(\vec{D}_{i,j}) \le -1.500 \\ 135°, & \text{if } -1.500 < \eta(\vec{D}_{i,j}) \le -0.668 \\ 157.5°, & \text{if } -0.668 < \eta(\vec{D}_{i,j}) \le -0.199 \\ 180°, & \text{if } -0.199 < \eta(\vec{D}_{i,j}) \le 0.414. \end{cases} \quad (5)$$

Then, the prediction pixel $(\tilde{p}_{i,j})$ is given as

$$\tilde{p}_{i,j} = \begin{cases} p_{i+1,j-1}, & \theta = 45° \\ p_{i,j-1}, & \theta = 90° \\ p_{i-1,j-1}, & \theta = 135° \\ p_{i-1,j}, & \theta = 180° \\ \text{round}((p_{i,j-1} + p_{i+1,j-1}) \div 2), & \theta = 67.5° \\ \text{round}((p_{i-1,j-1} + p_{i,j-1}) \div 2), & \theta = 112.5° \\ \text{round}((p_{i-1,j} + p_{i-1,j-1}) \div 2), & \theta = 157.5°. \end{cases} \quad (6)$$

The first four directions use copies of the pixels, whereas the others obtain the prediction by averaging two neighboring pixels. For the pixels in the last column, because the top-right reference pixel does not exist, the directions 45° and 67.5° are skipped.

In certain special cases, only one reference block may be available. For example, the first row pixels in the black rectilinear area have no top references, and the pixels in the first column in this area have no left ones. In these cases, we use the unique remaining block as the reference information to predict the edge direction, as shown in Fig. 2(b) and (c). Ten typical sequences (200 frames for each sequence) were adopted to verify the performance of our directional prediction algorithm. The statistics of the prediction error power in one 16 × 16 Luma partition are shown in Tables I and II. Compared with the original vertical/horizontal prediction, the prediction error energy is reduced by 54.1% on average. The performance gap between our directional prediction and the original horizontal prediction increases with increasing edge strength. When ES ≥ 10, the prediction accuracy can be improved by 20% with our proposed method. From Table III, we observed that the algorithm with directional intra prediction can achieve a 6.4% DRR increase on average.

TABLE I
PREDICTION PERFORMANCE ANALYSIS IN TERMS OF SUM OF SQUARED ERROR OF ONE 16 × 16 LUMA BLOCK

class	sequence	M0[1]	M1[2]	M2[3]	M3[4]
A	*PeopleOnStreet*	21244	10998	9561	9188
	Traffic	11134	6265	5803	5426
B	*ParkScene*	7983	6694	7286	6548
	Tennis	4715	1918	2234	1559
C	*BasketBallDrill*	24503	12889	11345	10191
	RaceHorses	30296	23580	22509	21658
D	*BasketballPass*	26808	14902	14153	12241
	BlowingBubbles	40326	22471	19767	18378
E	*Johnny*	12983	4118	3144	3115
	KristenAndSara	26789	9264	6941	6659
average		20678	11309	10274	9496

[1] original prediction method in [21]
[2] proposed prediction with left reference block
[3] proposed prediction with top reference block
[4] proposed prediction with both reference block

TABLE II
PREDICTION PERFORMANCE ANALYSIS IN TERMS OF PREDICTION ACCURACY OF ONE 16 × 16 LUMA BLOCK

class	sequence	Predicting Accuracy(%)					
		ES[1]<4		ES <10		ES ≥10	
		M0[2]	M1[3]	M0	M1	M0	M1
A	*PeopleOnStreet*	79.2	75.5	62.8	65.7	44.5	59.9
	Traffic	80.4	78.3	68.7	71.9	55.8	64.6
B	*ParkScene*	84.2	84.2	61.5	69.2	48.9	56.1
	Tennis	83.0	81.5	51.1	80.0	36.3	66.4
C	*BasketBallDrill*	69.2	69.2	55.0	62.5	38.6	61.4
	RaceHorses	70.5	67.7	54.5	63.6	36.9	60.1
D	*BasketballPass*	81.2	81.2	72.7	72.7	60.4	67.9
	BlowingBubbles	70.0	68.7	53.3	60.0	35.2	55.8
E	*Johnny*	93.0	93.9	58.6	82.8	32.5	66.3
	KristenAndSara	90.7	90.7	62.2	78.4	34.6	65.4
average		80.1	79.1	64.7	70.7	42.3	62.4

[1] $ES = dx^2{}_{i,j} + dy^2{}_{i,j}$, $dx_{i,j}$ and $dy_{i,j}$ represent the edge strength in the vertical and horizontal directions, respectively.
[2] M0: original prediction method in [21]
[3] M1: proposed directional prediction algorithm

B. Dynamic kth-Order Unary/Exp-Golomb Rice Coding

Our entropy coding algorithm is based on adaptive-order unary/Exp-Golomb rice coding. Specifically, with the provided order k, we define the quantization step as 2^k. Given the input value x, we have the quotient $q = x/2^k$ and the remainder $r = x\%2^k$. For the quotient part, we use the unary/Exp-Golomb coding with a cutoff value of 3. Specifically, when $q < 3$, unary coding [32] is applied; otherwise, we select Exp-Golomb coding [33], [34]. The remainder r is transmitted following the coded quotient. The coded examples with input values in the range of [0, 15] and the orders $k \in \{0, 1, 2\}$ are illustrated in Table IV. The underline separates the quotient part and the remainder part.

TABLE III
COMPRESSION PERFORMANCE (DRR) ANALYSIS OF DIRECTIONAL INTRA-PREDICTION SCHEME (LUMA ONLY, QP = 32)

class	Test Sequence	Directional Prediction	Horizontal Prediction
A	*PeopleOnStreet*	62.80%	57.71%
	Traffic	64.99%	60.59%
B	*BasketballDrive*	75.10%	72.17%
	BQTerrace	60.35%	47.12%
C	*BasketballDrill*	63.23%	57.32%
	BQMall	59.42%	48.78%
D	*BasketballPass*	64.94%	62.16%
	BQSquare	51.32%	41.40%
E	*Johnny*	74.76%	69.72%
	KristenAndSara	74.02%	69.77%
average		65.09%	58.67%

TABLE IV
UNARY/EXP-GOLOMB RICE CODING WITH ORDER $k \in \{0, 1, 2\}$

Coding residual	order k		
	k=0	k=1	k=2
0	0_	0_0	0_00
1	10_	0_1	0_01
2	110_	10_0	0_10
3	1110_	10_1	0_11
4	111100_	110_0	10_00
5	111101_	110_1	10_01
6	11111000_	1110_0	10_10
7	11111001_	1110_1	10_11
8	11111010_	111100_0	110_00
9	11111011_	111100_1	110_01
10	1111110000_	111101_0	110_10
11	1111110001_	111101_1	110_11
12	1111110010_	11111000_0	1110_00
13	1111110011_	11111000_1	1110_01
14	1111110100_	11111001_0	1110_10
15	1111110101_	11111001_1	1110_11
...	...	...	...

When $x \neq 0$, the sign bit is stored after the coded quotient part and the remainder part.

The high-order k possesses an advantage in coding the large value input. Given the input value x, the optimal order k_o, with which we can derive the maximum compression rate, is defined as (7). However, for the decoder side, the value of the current decode symbol x is unknown. Therefore, realizing the efficient pixel-grain k-order update is essential to our technique

$$\log_2(x/3) < k_o \leq \log_2(x/3) + 1. \tag{7}$$

Similar to the pixel value prediction method, we also use the correlations of order k among neighboring pixels. According to (4), we define four directions, and the order value $k_{i,j}$ of the current pixel is deduced from the previous decoded pixels as

$$k_{i,j} = \begin{cases} k'_{i-1,j}, & \text{if } -0.414 < \eta(\vec{D}_{i,j}) \leq 0.414 \\ k'_{i-1,j-1}, & \text{if } -2.414 < \eta(\vec{D}_{i,j}) \leq -0.414 \\ k'_{i,j-1}, & \text{if } |\eta(\vec{D}_{i,j})| > 2.414 \\ k'_{i+1,j-1}, & \text{if } 0.414 < \eta(\vec{D}_{i,j}) \leq 2.414. \end{cases} \tag{8}$$

In (8), the variable k' is derived using a fine adjustment to k. For the current position (i, j), we use $k_{i,j}$ to encode or decode the prediction residual $x_{i,j}$. Once the value of $x_{i,j}$ is obtained, we have the adjusted version of $k_{i,j}$, namely, $k'_{i,j}$, as described by (9). In this way, we realize the pixel-gain adaptive k-order. Because the maximum value of the order k is defined as 3, we need $2 \times 16 = 32$ bits to buffer the k' values in the upper row. Although we can use the previous decoder or encoder prediction residuals to adjust the current order, the buffer size is increased to $8 \times 16 = 128$ bits, which is four times that of our proposed order update scheme because the amplitudes of the residuals are in the range of [0, 255]. The value of $k_{0,0}$ is fixed and defined as 1, which outperforms other counterparts and provides an increase in the DRR of 0.2%. Table V shows the compression performance comparison of the dynamic order scheme, which achieved 4.2%–6.2% DRR increases compared with the fixed order

$$k'_{i,j} = \begin{cases} k_{i,j} + 1, & x \geq 3 \times 2^{k_{i,j}} \, (k_{i,j} < 3) \\ k_{i,j}, & 2^{k_{i,j}-1} \leq x < 3 \times 2^{k_{i,j}} \\ k_{i,j} - 1, & x < 2^{k_{i,j}-1} (k_{i,j} > 0). \end{cases} \tag{9}$$

TABLE V
COMPRESSION PERFORMANCE (DRR) ANALYSIS OF THE DYNAMIC ORDER SCHEME (LUMA ONLY, QP = 32)

class	Test Sequence	Dynamic Order	Order equal to 0	Order equal to 1
A	*PeopleOnStreet*	62.79%	57.57%	57.47%
	Traffic	64.99%	59.76%	59.37%
B	*BasketballDrive*	75.10%	74.07%	68.50%
	BQTerrace	60.35%	53.66%	53.37%
C	*BasketballDrill*	63.23%	60.20%	59.13%
	BQMall	59.42%	53.71%	53.95%
D	*BasketballPass*	64.94%	62.16%	60.74%
	BQSquare	51.32%	39.55%	40.82%
E	*Johnny*	74.76%	74.80%	68.31%
	KristenAndSara	74.02%	73.78%	67.33%
average		65.09%	60.93%	58.90%

C. Compression Skip Flag for Chroma Partition

It was observed that there were many continuous zeros in the chroma partitions with the proposed directional prediction method. Leveraging this property, we propose two compression skip flags (CSFs), i.e., the block CSF (BCSF) and the partition CSF (PCSF).

Specifically, one 8×8 chroma partition is evenly divided into eight 4×2 blocks. Each chroma block is assigned

TABLE VI
COMPRESSION PERFORMANCE (DRR) ANALYSIS OF THE CSF SCHEME (CHROMA ONLY, QP = 32)

class	Test sequence	with CSF	without CSF
A	*PeopleOnStreet*	82.23%	77.44%
	Traffic	82.03%	76.46%
B	*BasketballDrive*	85.55%	79.39%
	BQTerrace	87.01%	79.79%
C	*BasketballDrill*	75.98%	71.58%
	BasketballDrillText	75.10%	70.90%
D	*BasketballPass*	79.69%	75.21%
	BQSquare	84.77%	79.46%
E	*Johnny*	91.02%	82.03%
	vidyo3	92.87%	83.59%
average		83.63%	77.59%

a 1-bit dedicated CSF. When the BSCF is set, it indicates that the associated 4×2 chroma block is a zero-block. When all residuals in the 8×8 chroma partition are zeros, the 1-bit PCSF is set. With the quantization parameter (QP) being equal to 32, ten typical sequences in five different categories are tested to verify the performance of the CSF scheme. The compression results are shown in Table VI. The CSF scheme can enhance the DRR by 4.2%–9.3% for the chroma partitions. On average, the DRR increase originated from the use of CSF is 6.0%.

D. Hardware-Oriented Algorithm Simplification

Although the proposed algorithm achieves a high DRR, the associated hardware complexity is also increased. The directional intra-prediction module requires the great amount of hardware resources. Therefore, we optimize the edge strength calculation to simplify its implementation. For the $dx_{i,j}$ in (1a), the formula used to calculate the value is shown in the following equation and takes advantage of the carry save adder (CSA) architecture, as will be described in Section IV:

$$dx_{i,j} = p_{i-2,j} + p_{i-1,j} + \bar{p}_{i-2,j-1} + \bar{p}_{i-1,j-1} + 2. \quad (10)$$

When calculating the amplitude of the edge strength as (2), we need to compute $|dx_{i,j}|$. The rigorous formula of $|dx_{i,j}|$ is expressed as (11). In our design, we simplify (11) to (12), which saves one adder by sacrificing precision

$$|dx_{i,j}| = \begin{cases} dx_{i,j}, & dx_{i,j} \geq 0 \\ \bar{dx}_{i,j} + 1, & dx_{i,j} < 0 \end{cases} \quad (11)$$

$$|dx_{i,j}| = \begin{cases} dx_{i,j}, & dx_{i,j} \geq 0 \\ \bar{dx}_{i,j}, & dx_{i,j} < 0. \end{cases} \quad (12)$$

In addition, because the original parameter values in (5) are floating-point numbers, the primitive implementation will require a nontrivial chip area. Therefore, we transform the edge direction judgment from (5) to (13)

$$\theta = \begin{cases} 45°, & \text{if } 0.5|dx_{i,j}| < |dy_{i,j}| \leq 2|dx_{i,j}| \ \& \ s = 0 \\ 67.5°, & \text{if } 2|dx_{i,j}| < |dy_{i,j}| \leq 4|dx_{i,j}| \ \& \ s = 0 \\ 90°, & \text{if } |dy_{i,j}| > 4|dx_{i,j}| \\ 112.5°, & \text{if } 2|dx_{i,j}| \leq |dy_{i,j}| \leq 4|dx_{i,j}| \ \& \ s = 1 \\ 135°, & \text{if } |dx_{i,j}| \leq |dy_{i,j}| < 2|dx_{i,j}| \ \& \ s = 1 \\ 157.5°, & \text{if } 0.25|dx_{i,j}| \leq |dy_{i,j}| < |dx_{i,j}| \ \& \ s = 1 \\ 180°, & \text{if } (|dy_{i,j}| < 0.25|dx_{i,j}| \ \& \ s = 1) \\ & \text{or } (|dy_{i,j}| \leq 0.5|dx_{i,j}| \ \& \ s = 0) \end{cases} \quad (13)$$

in which, the parameter s indicates the sign of η and is calculated as

$$s = \text{sign}(dx_{i,j}) \oplus \text{sign}(dy_{i,j}). \quad (14)$$

TABLE VII
COMPRESSION PERFORMANCE (DRR) ANALYSIS OF THE VLSI-FRIENDLY EDGE STRENGTH AND EDGE DIRECTION CALCULATIONS

class	Test sequence	QP=22		QP=32	
		original	simplified	original	simplified
A	*PeopleOnStreet*	65.76%	65.79%	69.27%	69.27%
	Traffic	66.11%	66.21%	70.64%	70.67%
B	*ParkScene*	64.91%	64.88%	71.29%	71.22%
	Tennis	74.15%	74.22%	76.79%	76.79%
C	*BasketballDrill*	62.99%	62.27%	68.07%	67.48%
	RaceHorses	58.11%	57.98%	61.33%	61.20%
D	*BasketballPass*	65.10%	65.17%	69.82%	69.86%
	BlowingBubbles	51.92%	51.82%	57.39%	57.29%
E	*Johnny*	78.58%	78.55%	80.18%	80.18%
	KristenAndSara	77.83%	77.77%	79.23%	79.17%
average		66.55%	66.47%	70.40%	70.31%

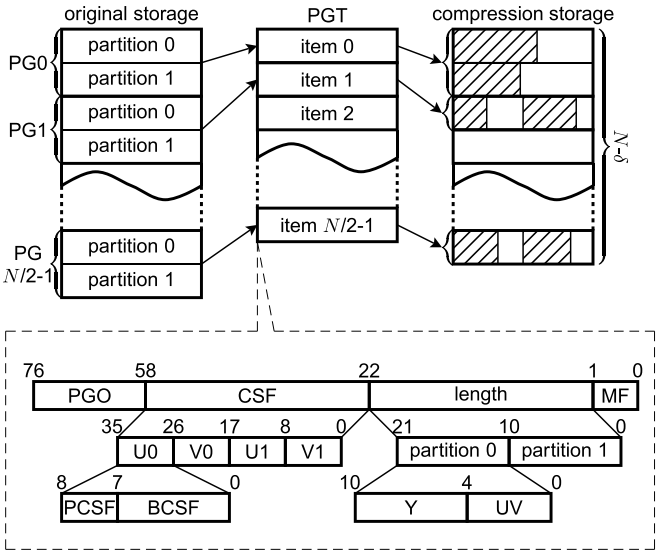

Fig. 4. Memory mapping for compression partition. (PGO used to record the beginning address of the PG. CSF consists of the PCSF and BCSF of the chroma components in a PG. Length: indicates the lengths of compressed luma and chroma components in the partition of the IO bitwidth, which help the decoder to optimize the burst read operation. MF used to indicate whether to use the compression storage scheme.)

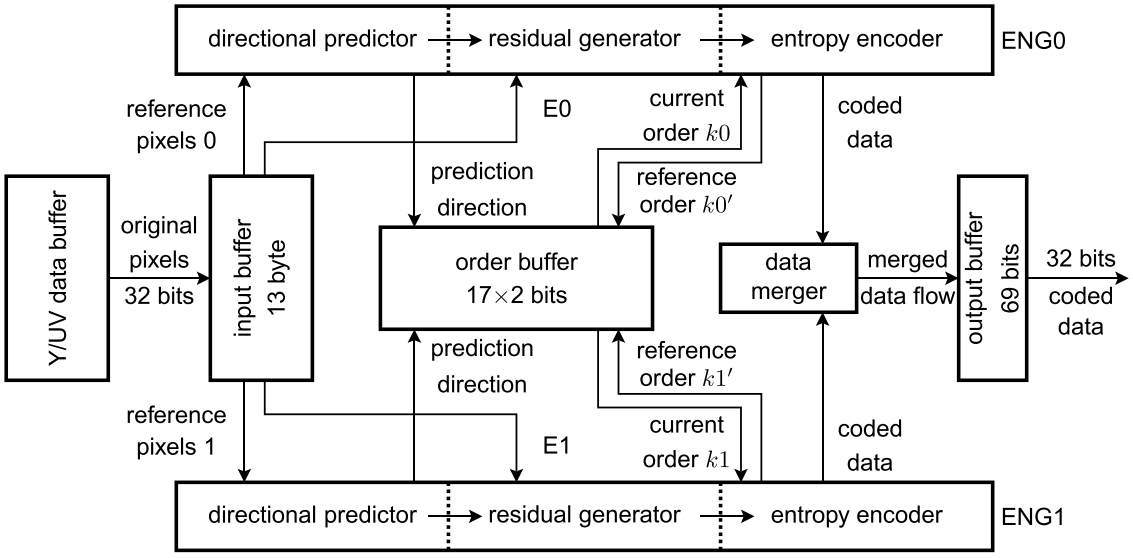

Fig. 5. Architecture of the proposed compressor (the locations of E0, E1, and reference pixels are shown in Fig. 6).

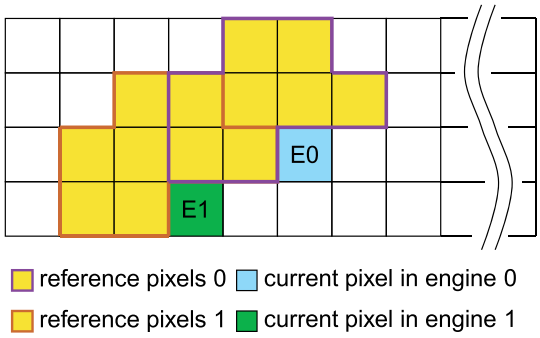

Fig. 6. Location of current and reference pixels.

We observe that all operations in (13) can be implemented with shifters and comparators.

Ten typical sequences in five different categories are tested to verify the performance of the hardware-friendly optimizations. The experiment results in Table VII show that the DRR degradation introduced by the hardware-friendly optimizations is only 0.08%–0.09% compared with the original counterpart.

III. Partition Group Table-Based Compression Storage

The over 60% average DRR of our algorithm makes it an efficient approach to reducing memory requirements and not simply reducing IO bandwidth requirements. However, the unfixed DRR of lossless compression cannot guarantee that the compressed partitions can be linearly addressed. In this paper, we use the PGT to handle the address mapping issues.

Specifically, every two horizontally adjacent 16 × 16 partitions constitute one partition group (PG), as shown in Fig. 4. Assuming that there are N partitions and that N is even, the PG number is $N/2$. Each PG possesses one dedicated PGT item to describe the properties of two leaf-node partitions. We can use one-partition space to store the content of one PG when the DRRs of the luma and chroma components of the two partitions are all no less than 50%; otherwise, the compressed PG will continue to consume two-partition storage spaces. In Fig. 4, PG0 represents the normal case and PG1 indicates the compression storage counterpart. In the compression storage, the second compressed partition data are always located in the lower half part. When the compression storage is adopted, the merge flag (MF) is set. If δ number of PGs can be stored in the compression mode, the memory space is reduced by δ/N. The experiments in Section V reveal that, on average, a memory savings of 38% is obtained.

The partition group offset (PGO) indicates the starting address of the PG in the partition length grain. Considering 8K resolution requirements, the bitwidth of PGO is defined as 18 bits. To help the decoder side determine the optimal burst length, we also provide the length information in the PGT. As will be explained in Section IV, although the luma pixels in odd and even rows are encoded and decoded in parallel, the coded data of the odd and even rows are merged to be alternately stored. Therefore, only one luma length is required. It should be noted that our algorithm cannot guarantee that the compressed data length is less than the source one. When the compression ratio is no greater than 1, we store that original data. When the length value is equal to the original one, the decoder indicates that the stored data should use the normal pixel format.

By adopting the storage compression method, our work not only reduces the storage size but also decreases the frequency of precharge and activate operations during external DRAM accessing. According to [10], the power consumed by row buffer activation accounts for 38% of the total dynamic power of the DRAM. Therefore, improving the utilization of row buffer data by our compression method is an efficient approach to reducing the external DRAM power requirements. The detailed power saving analysis of the external DRAM will be described in Section V.

TABLE VIII
COMPARISON RESULTS OF THE THREE ARCHITECTURES OF THE INTRA-PREDICTION OPERATOR

	Original	CMPR42 [35]	CSA
Gate count (K)	4.9	6.3	1.9
Max freq. (MHz)	523	563	578

IV. HARDWARE IMPLEMENTATION

A. Compressor Implementation

Fig. 5 provides the top block diagram of the proposed compressor with the directional intra prediction and the k-order UEG-Rice coding. The input signals are 16×16 partitions of reconstructed pictures, input from the encoder or the decoder.

To improve the throughput, a two-engine architecture is devised. For the 16×16 luma partition, the pixels in the odd and even rows are processed in parallel. Because the order k value of the top-right pixel is required when encoding the current pixel, we adopt the wavefront mode. Specifically, the compression process of the previous row should be at least two pixels in advance of that of the current row. On the other hand, because a data dependency does not exist between the U and V partitions, the U and V 8×8 partitions can be simultaneously encoded. The coded data of the odd and even rows are merged into one stream and stored in the output buffer. If the length of the output buffer is greater than 32 bits, the compressor will directly write its output to the memory controller. The reference pixel buffer is shared by two encoder engines. Because the ENG0 is strictly 2 pixels in advance of ENG1, the scale of the reference pixel buffer is 13 B, as shown in Fig. 6.

The order k in UEG-Rice coding requires the previous k' values to be buffered. The primitive implementation, stores the k' values in the previous three rows, which accounts for $16 \times 3 \times 2 = 96$ bits. In our design, by strictly scheduling the two engines, we can discard the k' values in ENG0 that are no longer required by ENG1, and the freed memory space can be used by ENG1 for its k' value storage. Consequently, the buffer size is reduced to $17 \times 2 = 34$ bits.

To increase the clock speed, the encoder engine is composed of three pipeline stages: 1) the directional predictor; 2) residual generator; and 3) entropy encoder.

The primitive design of $|dx^{l}_{i,j}|$, as shown in Fig. 7(a), consumes four adders, four subtracts, one comparator, and one multiplexer. To improve the clock speed, a 4-2 compressor-based absolute difference computation method was proposed in [35] and shortens the critical paths by parallel processing. The shortcoming of this method lies on the additional hardware cost for the parallelism. Our paper proposes the hardware-friendly algorithm in (12), and the corresponding circuits design is shown in Fig. 7(c). By sacrificing 1 bit of precision, the circuit's performance is significantly improved. A comparison of the primitive, the 4-2 compressor based, and our proposed circuits is shown in Table VIII. The results show that the proposed architecture achieved a decrease in area of 61.2% and an increase in frequency of 10.5% compared with the primitive one.

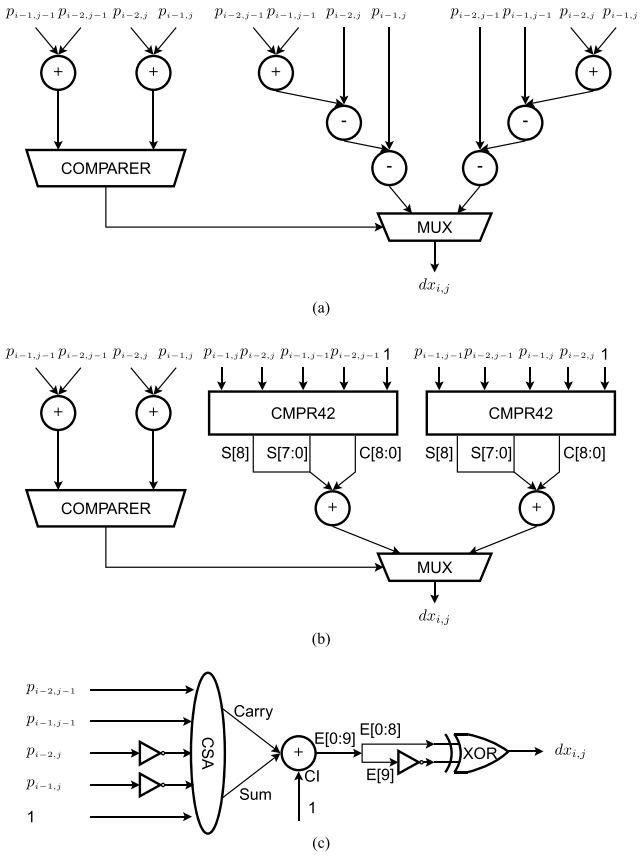

Fig. 7. Hardware implementation for calculating $|dx_{i,j}|$ ($p_{i-2,j-1}$, $p_{i-1,j-1}$, $p_{i-2,j}$, $p_{i-1,j}$ are neighboring reference pixels, as shown in Fig. 2). (a) Primitive architecture. (b) 4-2 compressor based architecture. (c) Proposed architecture.

B. Decompressor Implementation

Fig. 8 depicts the top block diagram of the decompressor. Based on the index information from the PGT, the decompressor derives the begin address, the length, and the CSF of the luma and chroma partitions. The begin address and the length values guide the memory controller to adopt the optimal read mode and burst length to fetch the compressed data from the system DRAM. The neighboring pixels are required to perform the directional prediction during the decompression process. We can observe from Fig. 6 that four rows of pixels should be stored; therefore, the sizes of the two SRAM are 32 bit $\times$ 16 word.

The decompressor employs a three-stage pipeline, including data fetch, entropy decoding, and directional prediction. The horizontal data dependency between neighboring pixels will seriously degrade the pipeline utilization with the primitive scheduling, as shown in Fig. 9(a). We observe that every three cycles, only one pixel is decoded. The hardware utilization is only 33%. Because the dependency of the odd and even rows in Y can be ameliorated with the wavefront decoding mode and because dependencies among luma and chroma partitions do not exist, we apply the interchange decoding of Y odd row, Y even row and UV components to increase the pipeline efficiency, as described by Fig. 9(b). With the proposed schedule, the hardware utilization is improved to up to 100%.

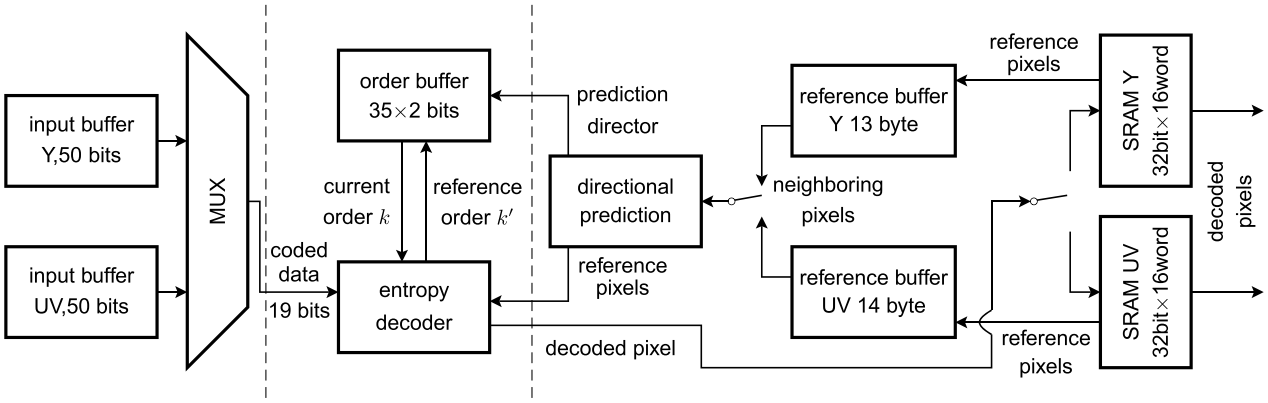

Fig. 8. Architecture of the proposed decompressor.

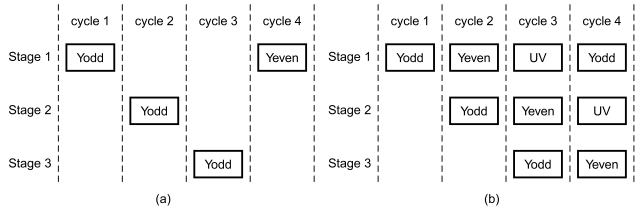

Fig. 9. Processing schedule of (a) primitive decompression architecture and (b) proposed YUV interchange decompression architecture.

The top block diagram of the HEVC encoder integrated with the IO codec module is shown in Fig. 10. To obtain a high throughput, we apply the two-parallel-codec structure to simultaneously handle two partitions in one PG. To address the storage space compression, SRAM is applied to cache the encoded bitstream of the second partition. Specifically, the space compression is in PG granularity. During the encoding, the first engine directly writes its output with a 32-bit grain to the memory controller, because its begin address has been determined. However, the second engine needs to write its output stream to the SRAM. For the writing from the SRAM to the memory controller, there are two situations that must be discussed.

1) During the encoding procedure, if either DRR of the two engines is less than 50%, then the begin address of the second engine is determined. Thereafter, the memory controller can fetch the buffered data in the SRAM, and the empty entries are left for the new coded stream.
2) At the end of the encoding of one PG, if two partitions can share the same partition space, then the data in the SRAM can be dispatched to the memory controller. Therefore, the minimum volume of the SRAM is 32 bit × 48 word, which is half of the original partition size.

The memory controller [36] uses a read reorder buffer [37]. Level-D search region data reuse is adopted in our design [7]. The reference frame buffer size is FW × ($\mathrm{SR}_V + N - 1$), as shown in Fig. 10. When encoding the Coding Tree Units (CTUs) in the next row ($\mathrm{CTU}_{i,j+1}$), the pixels in the overlap area can be reused. It is merely required to load the data in the emerald area to replace the content in the yellow rectangle. In this way, assuming that the sampling format is 4:2:0 and that the reference frame number is 1, the IO bandwidth requirements are reduced to

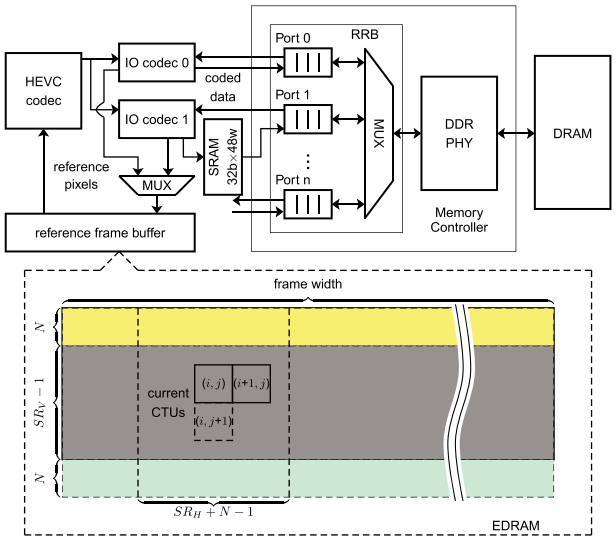

Fig. 10. Block diagram of the two-parallel-codec structure. (RRB: read reorder buffer, SR_V: the search range in the vertical direction, SR_H: the search range in the horizontal direction, and N: the size of the CTU.)

1.5 × FW × FH × FR pixels/s, where FW and FH represent the width and height of the picture and FR denotes the frame rate. For example, with 4K @ 60 frames/s and a 4:2:0 sampling format, the IO bandwidth for search region loading is 712 MB/s. The main hindrance of level-D data reuse strategy lays in the large on-chip memory requirements. Using the embedded SRAM to implement the level-D reference buffer is area and power intensive. In contrast, the embedded DRAM (eDRAM) possesses the advantages of higher density and lower static power compared with the SRAM [38], [39]. Because the primary IC foundries, such as TSMC and United Microelectronics Corporation, have provided the high-performance eDRAM IPs [40], [41], eDRAM is becoming a common design tool for advanced processors and application-specified integrated circuit design [42]–[45]. In our design, eDRAM is employed to implement the level-D search region buffering.

V. Experimental Results

A. Compression Performance Analysis

The proposed frame memory compression algorithm is integrated with HM12.0 reference software to evaluate its performance. A total of 24 typical video sequences belonging

TABLE IX
DRR OF PROPOSED ALGORITHM AND PREVIOUS LOSSLESS ALGORITHM (QP = 37, 200 FRAMES, GOP = IBBB)

class	video sequence	proposed algorithm(%)	DPCM [21](%)
A	*PeopleOnStreet*	70.64	59.90
	Traffic	72.10	61.20
B	*crowd_run*	61.91	51.30
	BasketballDrive	79.39	68.23
	BQTerrace	70.80	54.43
	Cactus	73.05	61.72
	Kimono1	75.68	65.10
	ParkScene	73.96	60.94
	Tennis	77.96	63.28
C	*BasketballDrill*	69.27	58.33
	BasketballDrillText	68.91	57.81
	BQMall	66.89	52.86
	PartyScene	55.14	43.49
	RaceHorses	64.06	52.86
D	*BasketballPass*	73.21	61.72
	BlowingBubbles	60.16	48.18
	BQSquare	64.23	53.59
	RaceHorses	61.59	50.00
E	*Johnny*	80.83	70.05
	KristenAndSara	79.75	68.75
	SlideEditing	70.28	58.85
	vidyo1	79.17	66.67
	vidyo2	80.24	68.23
	vidyo3	81.05	68.49
average		71.25	59.31

TABLE X
AVERAGE DRR COMPARISON (LUMA ONLY)

QP	DRR(%)			
	Kim's [22]	Cheng's [19]	Guo's [27]	Proposed
22	48.87	50.15	54.22	58.21
27	51.49	52.76	57.16	60.63
32	53.65	54.43	58.88	62.62
37	55.22	55.76	60.12	64.60
Avg	52.36	53.28	57.60	61.52

to five classes were tested. All sequences were encoded with 200 frames, and IBBB GOP was used.

The performance comparisons in terms of the DRR of the proposed algorithm and the DPCM lossless algorithm [21] with QP = 37 are shown in Table IX. The compression efficiency of our proposed methods outperformed the previous method in all benchmarks. The average DRR increase achieved by our method is 11.9% compared with the DPCM counterpart. The average performance comparisons in terms of the DRR of the proposed algorithm and the other three lossless algorithms are shown in Table X. The proposed algorithm can achieve an increase in the DRR of at least 3.9% compared with the other algorithms.

The HEVC profile Main12 supports bit depths beyond 8 bits/sample. The extension of the bit depths makes HEVC well suited to UHD TV, where very high video quality is essential. In HM software, the storage of one 10-bit sample (Y, Cb, or Cr) consumes 16-bit of space. Therefore, the increase in bit-depth wastes a substantial amount of memory space and decreases the external bandwidth utilization. We can see that, for the high-bit-depth videos, if the average bit depth of the color channels can be reduced to 8 bit, one 16×16 partition can be saved in a $(16 \times 16 + 8 \times 8 \times 2)\times$ 8-bit area, instead of occupying a $(16 \times 16 + 8 \times 8 \times 2) \times$ 16-bit region. The performance of DRAM can be significantly improved. Compared with the previous lossless compression algorithms, our method provides two types of advantages: first, the energy of the residuals is reduced by 23.2%–56.4% using the precision directional prediction, as shown in Table XI. Second, the k-order UEG-Rice coding efficiently codes the large-valued prediction residues. Our experiments in Table XII reveal that 74.5%–92.0% of the partitions in the 30 bpp video sequences can be compressed to the average 8-bit bit depth, 18.7%–56.1% higher than the predecessor [21]. On average, 85.6% of the partitions can be stored in half of the space that they previously occupied. Therefore, a reduction in the activate power of 42.8% can be obtained. In addition, we compared the compression performance of the proposed algorithm with the HEVC lossless method [46], and the results are shown in Table XIII. For the high-resolution video sequences (classes A, B, and E), the proposed algorithm improved the DRR by 8.7%–12.7% compared with the method in [46]. For the 30 bpp sequences, the performance gap was increased to 13.5%.

The compression also leads to a reduction in the dynamic power consumption of the DRAM. The power performance is

TABLE XI
SUM OF SQUARED ERROR OF ONE 16×16 LUMA BLOCK FOR VIDEO SEQUENCES WITH A BIT DEPTH OF 10 bits/SAMPLE

video sequence	QP	DPCM [21]	proposed	reduction
NebutaFestival	22	969349	545669	43.7%
	37	711146	310377	56.4%
SteamLocomotiveTrain	22	328795	252401	23.2%
	37	254099	176121	30.7%

TABLE XII
COMPARISONS OF PROPOSED AND DPCM ALGORITHMS IN COMPRESSING 30 bpp SEQUENCES [THE METRIC IS THE RATIO OF 16×16 PARTITIONS THAT CAN BE COMPRESSED IN $(16 \times 16 + 8 \times 8 \times 2) \times$ 8-bit MEMORY REGIONS]

video sequence	QP	proposed algorithm(%)	DPCM [21](%)
NebutaFestival	22	74.47	18.33
	27	77.72	24.35
	32	84.65	48.96
	37	88.71	62.50
SteamLocomotiveTrain	22	87.86	65.21
	27	89.02	67.44
	32	90.12	69.68
	37	91.97	73.28

TABLE XIII
DRR OF THE PROPOSED ALGORITHM AND THE HEVC LOSSLESS METHOD [GOP = IBBB, QP = (22, 27, 32, 37)]

class	HEVC method [46]	proposed algorithm
A	58.5%	68.9%
B	57.6%	70.3%
C	60.3%	61.7%
D	62.1%	60.4%
E	68.7%	77.4%
30 bpp	33.8%	47.3%

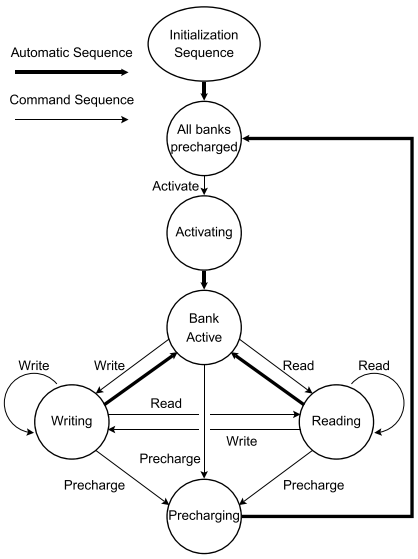

Fig. 11. Simplified DRAM data burst reading/writing state diagram.

measured by the CACTI simulator [47] with the 1.5 V core and IO voltages. The state-transition diagram of the DRAM is given by Fig. 11. The DRAM is organized with a 2-D array structure. The entire data array is composed of multiple identical banks, which can be simultaneously accessed with different data buses. A row is simply a group of memory cells that are activated in response to a row activation command. A column of data is the smallest independently addressable unit of memory, and its size is identical to the output data width. Each column access reads or writes multiple columns of data depending on the burst length. When the desired data are not located in the DRAM row buffer, the precharge and activate operations, which save the current row content to the memory bank and then reload the desired data row from the memory bank, are dispatched. The precharge and activate operations account for 38.2% of the overall dynamic power requirements of the DRAM. Our memory space compression algorithm reduced the number of precharge and activate operations by on average 38%. Accordingly, the dynamic power dissipation of the DRAM is reduced by 14.5% in our experiments.

B. Hardware Implementation Analysis

The hardware architecture is described in Verilog HDL and synthesized with TSMC 65-nm standard cell libraries, using Synopsys Design Compiler and IC-Compiler to obtain accurate postlayout timing, area, and power estimation. Table XIV shows the hardware implementation results for the proposed compressor and decompressor.

TABLE XIV
HARDWARE IMPLEMENTATION RESULTS FOR THE COMPRESSOR AND DECOMPRESSOR (VOLTAGE = 0.9 V AND TEMPERATURE = 125 °C)

	Compressor	Decompressor
CMOS technology	65nm	
Gate count(K)	36.5	34.7
On-chip SRAM(byte)	192	256
Max freq.(MHz)	578	599
Power consumption(mW)	5.3	5.0

In the worst working conditions (0.9 V, 125 °C), the compressor achieved 578-MHz clock speed with only 36.5k-gate standard cells and 192-B SRAM, while the decompressor consumed 34.7k-gate standard cells and 256-B SRAM at the frequency of 599 MHz. The power consumption of the compressor and decompressor are 5.3 and 5.0 mW, respectively.

Table XV shows a comparison of the compressor implementation of the proposed algorithm with previous lossy and lossless algorithms. In Lee's lossy algorithm [12], the DRR was fixed as 50%; therefore, the memory bandwidth as well as the memory requirements could be reduced by half. Because it is a lossy algorithm, there is a gradual decrease in quality because of error propagation. Therefore, the lossy algorithm is not suitable for the HEVC encoder.

The other algorithms are lossless algorithms. The work in [19] proposed a multimode embedded compression algorithm based on set-partitioning in hierarchical trees (SPIHT) and achieved a 59.6% DRR. The complex circuit design seriously degraded the clock speed to 10 MHz. The low throughput (4.5 Mpixels/s) prevents this algorithm from being used in HD/UHD coding scenarios. The algorithm in [22] consists of a hierarchical prediction method based on pixel averaging and pixel copying and SBT. The high DRR (57.3%) mainly originates from the accurate hierarchical prediction method. On the other hand, because four pixels in one 8×8 partition could be processed in parallel, the throughput was as high as 0.92 Gpixels/s. A variable-length coding based on DPCM was proposed in [21] and could achieve a 56.9% DRR. It achieved a throughput of 0.53 Gpixels/s through two-engine parallelism. A lossless reference frame recompression algorithm based on an MDA prediction scheme and semifixed length coding was proposed in [27]. This algorithm achieved an average DRR of 61.9%, and the throughput was as high as 3.13 Gpixels/s. Although the previous lossless algorithms contributed to the IO traffic reduction, they ignored the memory size optimization.

The proposed algorithm obtained a DRR of 68.5%, which is 6.6%–18.5% higher than the predecessors. Because we introduced the three-stage pipeline architecture, the maximum clock speed under the worst conditions is 578 MHz for the compressor, which is 1.93–57.8 times that of

TABLE XV
COMPARISONS OF THE IMPLEMENTATION OF THE PROPOSED ALGORITHM WITH PREVIOUS LOSSY AND LOSSLESS ALGORITHMS

	Lee's [12]	Cheng's [19]	Kim's [22]	Zhou's [21]	Guo's [27]		Proposed	
	comp.	comp.	comp.	comp.	comp.	decomp.	comp.	decomp.
Data reduction ratio(%)	50.0(fixed)	59.6	57.3	56.9	61.9		68.5	
CMOS technology	0.18μm	0.18μm	0.18μm	90nm	90nm		65nm	
Gate count(K)	28.0	26.9	36.1	N/A	45.1	34.5	36.5	34.7
On-chip SRAM(byte)	0	512	0	N/A	N/A		192	256
Max freq.(MHz)	14	10	180	175	300		578	599
Throughput(pixels/cycle)	2.6	0.45	5.1	3	10.7	21.3	2.67	1.33
Throughput(Gpixels/s)	0.036	0.0045	0.92	0.53	3.13	6.26	1.54	0.78
Memory space save(%)	50	0	0	0	0		38	
Dynamic power save(%)	50.0	36.8	35.4	35.1	38.2		56.8	
ΔPSNR(db)	-0.12	0	0	0	0		0	
Coding method	Golomb-Rice	SPIHT	SBT	DPCM&SFL	MDA&SFL		URG-Rice	
Compression type	Lossy	Lossless	Lossless	Lossless	Lossless		Lossless	

other algorithms. Because of the two-parallel-codec structure, the maximum encoding throughput is 1.54 Gpixels/s, and the decoding throughput is 0.78 Gpixels/s. The primary advantage of the proposed architecture is that it can reduce memory requirements in the same manner as the lossy algorithm. On average, 38% memory reductions were achieved, which is only 12% lower than the lossy counterpart. The internal read/write power, the IO terminal power and the activate power consume 40.1%, 21.7%, and 38.2% of the total dynamic power, respectively. Therefore, for both reductions of the IO traffic and the frequency of precharge and activate operations, the dynamic energy consumption of DRAM can be reduced by 56.8% on average.

VI. CONCLUSION

To reduce the dynamic power requirements of DRAM in the video codec system, this paper proposes a lossless compression algorithm that reduces the external traffic and storage requirements of the reference frames. First, pixel-granularity adaptive directional prediction is adopted to reduce the prediction residual energy. Second, dynamic kth-order unary/Exp-Golomb rice coding is applied to accommodate the large-valued prediction residuals. The experimental results demonstrate that the proposed algorithm reduced the OFF-chip data traffic by 68.5% on average. By applying the PGT-based storage space compression scheme, we can further reduce the memory requirements by 38%. Because of the IO traffic reduction and row buffer utilization improvement, a total of 56.8% of the dynamic power of external DRAM can be saved by our strategies. Based on TSMC 65-nm CMOS technology, our parallel compressor and decompressor achieved the peak-age throughputs of 1.54 and 0.78 Gpixel/s, respectively, which can handle QHFD (4K) @ 94 frames/s real-time encoding by applying the level-D reference data reuse scheme.

REFERENCES

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[2] M. Tikekar, C.-T. Huang, C. Juvekar, V. Sze, and A. P. Chandrakasan, "A 249-Mpixel/s HEVC video-decoder chip for 4K ultra-HD applications," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 61–72, Jan. 2014.

[3] *4K Resolution*. [Online]. Available: http://en.wikipedia.org/wiki/4K_resolution, accessed Oct. 17, 2015.

[4] *8K Resolution*. [Online]. Available: http://en.wikipedia.org/wiki/8K_resolution, accessed Oct. 14, 2015.

[5] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[6] T.-Y. Oh *et al.*, "A 7 Gb/s/pin 1 Gbit GDDR5 SDRAM with 2.5 ns bank to bank active time and no bank group restriction," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 107–118, Jan. 2011.

[7] C.-Y. Chen, C.-T. Huang, L.-G. Chen, and L.-G. Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 553–558, Apr. 2006.

[8] O. Vargas, "Achieve minimum power consumption in mobile memory subsystems," *EE Tines Asia*, Mar. 2006.

[9] T. Nishikawa *et al.*, "A 60 MHz 240 mW MPEG-4 video-phone LSI with 16 Mb embedded DRAM," in *IEEE Int. Solid-State Circuits Conf. (ISSCC), Dig. Tech. Papers*, Feb. 2000, pp. 230–231.

[10] Micron Company, "Calculating memory system power for DDR3," Micron Technol., Inc., Boise, ID, USA, Tech. Rep. TN-41-01, 2007.

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

[12] 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.

[13] A. D. Gupte, B. Amrutur, M. M. Mehendale, A. V. Rao, and M. Budagavi, "Memory bandwidth and power reduction using lossy reference frame compression in video encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 225–230, Feb. 2011.

[14] N. Nazari, R. Shams, M. Mohrekesh, and S. Samavi, "Near-lossless compression for high frame rate videos," in *Proc. 21st Iranian Conf. Elect. Eng. (ICEE)*, May 2013, pp. 1–6.

[15] F. Sampaio, B. Zatt, M. Shafique, L. Agostini, J. Henkel, and S. Bampi, "Content-adaptive reference frame compression based on intra-frame prediction for multiview video coding," in *Proc. 20th IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2013, pp. 1831–1835.

[16] L. Santos, S. López, G. M. Callicó, J. F. López, and R. Sarmiento, "Performance evaluation of the H.264/AVC video coding standard for lossy hyperspectral image compression," *IEEE J. Sel. Topics Appl. Earth Observat. Remote Sens.*, vol. 5, no. 2, pp. 451–461, Apr. 2012.

[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 Trans. Circuits Syst. Video Technol.*, vol. 25, no. 1, pp. 112–124, Jan. 2015.

[18] Z. Shen, C. Miao, and Y. Zhang, "Memory bandwidth reduction for video decoders based on data arrangements," in *Proc. 6th Int. Congr. Image Signal Process. (CISP)*, vol. 1. Dec. 2013, pp. 31–35.
[19] 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.
[20] D. Silveira, G. Povala, L. Amaral, B. Zatt, L. Agostini, and M. Porto, "An energy-efficient hardware design for lossless reference frame compression in video coders," in *Proc. IEEE 20th Int. Conf. Electron., Circuits, Syst. (ICECS)*, Dec. 2013, pp. 573–576.
[21] D. Zhou *et al.*, "A 530 Mpixels/s 4096×2160@60 fps H.264/AVC high profile video decoder chip," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 777–788, Apr. 2011.
[22] J. Kim and C.-M. Kyung, "A lossless embedded compression using significant bit truncation for HD video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 6, pp. 848–860, Jun. 2010.
[23] G. Povala, D. Silveira, L. Amaral, B. Zatt, M. Porto, and L. Agostini, "An efficient reference frame compression approach for video coding systems," in *Proc. IEEE 5th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Feb. 2014, pp. 1–4.
[24] H.-C. Kuo and Y.-L. Lin, "A hybrid algorithm for effective lossless compression of video display frames," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 500–509, Jun. 2012.
[25] G.-L. Li, Y.-C. Chen, Y.-H. Liao, P.-Y. Hsu, M.-H. Wen, and T.-S. Chang, "A 135 MHz 542 k gates high throughput H.264/AVC scalable high profile decoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 4, pp. 626–635, Apr. 2012.
[26] Q. Cai, L. Song, G. Li, and N. Ling, "Lossy and lossless intra coding performance evaluation: HEVC, H.264/AVC, JPEG 2000 and JPEG LS," in *Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Dec. 2012, pp. 1–9.
[27] L. Guo, D. Zhou, and S. Goto, "A new reference frame recompression algorithm and its VLSI architecture for UHDTV video codec," *IEEE Trans. Multimedia*, vol. 16, no. 8, pp. 2323–2332, Dec. 2014.
[28] K. McCann, B. Bross, W.-J. Han, I.-K. Kim, K. Sugimoto, and G. J. Sullivan, *High Efficiency Video Coding (HEVC) Test Model 12 (HM12) Encoder Description*, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding, document JCTVC-N1002, Jul. 2013.
[29] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[30] 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.
[31] F. Pan *et al.*, "Fast mode decision algorithm for intraprediction in H.264/AVC video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 813–822, Jul. 2005.
[32] S. Xue and B. Oelmann, "Unary prefixed Huffman coding for a group of quantized generalized Gaussian sources," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1164–1169, Jul. 2006.
[33] T. Silva, J. Vortmann, L. Agostini, S. Bampi, and A. Susin, "FPGA based design of CAVLC and Exp-Golomb coders for H.264/AVC baseline entropy coding," in *Proc. 3rd Southern Conf. Program. Logic (SPL)*, Feb. 2007, pp. 161–166.
[34] S. Nargundmath and A. Nandibewoor, "Entropy coding of H.264/AVC using Exp-Golomb coding and CAVLC coding," in *Proc. Int. Conf. Adv. Nanomater. Emerg. Eng. Technol. (ICANMEET)*, Jul. 2013, pp. 607–612.
[35] H. Kaul *et al.*, "A 320 mV 56 μW 411 GOPS/watt ultra-low voltage motion estimation accelerator in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 107–114, Jan. 2009.
[36] *DesignWare Enhanced Universal DDR Memory Controller IP (uMCTL2) Datasheet*, Synopsys, Inc., Mountain View, CA, USA, 2015.
[37] *Achieve 10X DRAM Bandwidth Improvement With a DDR Controller Read Reorder Buffer*, Synopsys, Inc., Mountain View, CA, USA, 2013.
[38] *eDRAM*. [Online]. Available: http://en.wikipedia.org/wiki/EDRAM, accessed Sep. 2, 2015.
[39] S. Mittal, J. S. Vetter, and D. Li, "A survey of architectural approaches for managing embedded DRAM and non-volatile on-chip caches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1524–1537, Jun. 2015.
[40] *High Density Memory*. [Online]. Available: http://www.tsmc.com/english/dedicatedFoundry/technology/hdm.htm, accessed 2014.
[41] *UMC's Embedded DRAM, URAM Proven in 65 nm Customer Silicon*. [Online]. Available: http://www.umc.com/English/news/2008/20080804.asp, accessed Aug. 2008.
[42] Y. S. Park, D. Blaauw, D. Sylvester, and Z. Zhang, "Low-power high-throughput LDPC decoder using non-refresh embedded DRAM," *IEEE J. Solid-State Circuits*, vol. 49, no. 3, pp. 783–794, Mar. 2014.
[43] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2014, pp. 609–622.
[44] R. Kalla, B. Sinharoy, W. J. Starke, and M. Floyd, "Power7: IBM's next-generation server processor," *IEEE Micro*, vol. 30, no. 2, pp. 7–15, Mar./Apr. 2010.
[45] *Xbox 360*. [Online]. Available: http://en.wikipedia.org/wiki/Xbox_360, accessed Oct. 24, 2015.
[46] M. Zhou, W. Gao, M. Jiang, and H. Yu, "HEVC lossless coding and improvements," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1839–1843, Dec. 2012.
[47] *CACTI: An Integrated Cache and Memory Access Time, Cycle Time, Area, Leakage, and Dynamic Power Model*. [Online]. Available: http://www.hpl.hp.com/research/cacti/, accessed 2008.

Xiaocong Lian (S'15) received the B.S. degree in electronic science and technology from Northwestern Polytechnical University, Xi'an, China, in 2013.

His research interests include algorithms and very-large-scale integration implementation for video coding.

Zhenyu Liu (M'07) received the B.E., M.E., and Ph.D. degrees in electrical engineering from Beijing Institute of Technology, Beijing, China, in 1996, 1999, and 2002, respectively.

He held a post-doctoral position with Tsinghua University, Beijing, from 2002 to 2004, where he concentrated on the embedded processor architecture design. From 2004 to 2009, he was a Visiting Researcher with the Graduate School of Information, Production and Systems, Waseda University, Tokyo, Japan. He is currently an Associate Professor with Tsinghua University. His current research interests include low-power algorithm, SoC design for video compression, such as H.264/AVC and H.265/HEVC, and architecture research of application-oriented many-core processor design.

Wei Zhou (M'11) received the B.E., M.S., and Ph.D. degrees from Northwestern Polytechnical University, Xi'an, China, in 2001, 2004, and 2007, respectively.

He is an Associate Professor with Northwestern Polytechnical University. His research interests include video coding and associated VLSI architecture design.

Zhemin Duan received the B.E. and M.S. degrees from Northwestern Polytechnical University, Xi'an, China, in 1978 and 1983, respectively.

He is a Professor with Northwestern Polytechnical University. His research interests include video coding and associated VLSI architecture design.