A Novel Embedded Bandwidth-Aware Frame Compressor for Mobile Video Applications

Yu-De Wu
Department of Electronic Engineering
NCTU
Hsinchu, Taiwan
krus@si2lab.org

Yao Li
Department of Electronic Engineering
NCTU
Hsinchu, Taiwan
yaoli@si2lab.org

Chen-Yi Lee
Department of Electronic Engineering
NCTU
Hsinchu, Taiwan
cylee@si2lab.org

***Abstract*-- To reduce the bandwidth requirement and the size of frame memory for video decoding, embedding a compressor/decompressor on the chip is a well-known solution. Video compression has been developed for a long time and numbers of algorithm have been proposed. Those algorithms show us that enhancing the complexity can reach better performance. However, algorithm with higher complexity is more difficult to be embedded in a video decoder system. Longer coding cycles and lager gate counts may become a heavy load of the original system. In this paper, a new embedded compressor/decompressor algorithm is proposed for H.264 video compression. It is a lossy compression formed by two dimensions discrete cosine transform and modified bit plane zonal coding. The proposed algorithm compresses the 4x4 size block into a 64 bits segment, and decompresses a frame into 4x4 blocks. The compression ratio is fixed at two. With pipelined architecture, it takes 72 cycles and 34 cycles per MB for encoding and decoding respectively. As a result, our proposal becomes flexible to be embedded with any video coding system to save power consumption while maintaining acceptable video quality.**

I. INTRODUCTION

To improve the video coding efficiency, eliminating temporal redundancy between frames is a useful technique. This technique is widely used in nowadays video coding standards such as MPEG-1/2/4, H.263 and H.264. But to accomplish this method when encoding or decoding, at least one previous frame must be stored in frame memory as reference. Also, when processing motion compensation function in H.264 decoder, the rapid data accesses dominate the power consumption of whole system. For a mobile device, power is always the critical issue that people do care about. Embedded compression (EC) is a common technique to reduce the transferring of data and the size of off-chip frame memory. Moreover, if we embedded a compressor into a system with determined bandwidth, the access times can be efficiently reduced as long as the compressed unit is well-designed. Nowadays, the mobile devices become more and more powerful by their various functions, to reduce the bandwidth and resource requirement of each hardware accelerator is definitely an important topic.

Basically, compression can be divided into two types, lossy and lossless. Lossless methods are good at quality but suffered from variable data amount after compressed. Variable data amount cannot guarantee the reduction neither on the size of external frame memory nor the bandwidth. Lossy compression technique is suitable here because lossy compression with fixed compression ratio can ensure the reduction. Therefore, how to organize the lossy coding methods is important. To cover information as much as possible within limited budget is the main challenge. Several research activities about embedded compression have been proposed in [2-4]. Discrete cosine transform with high efficiency bit plane zonal coding have been proposed in [1], this JPEG-like methods provides good compressed quality, and the hardware is relatively smaller than JPEG. But the bit plane zonal coding is too complicate, thus its processing cycles may become too long and not suitable for being embedded with H.264 video decoder. Also, the input data of motion compensation is provided through the embedded decompressor. The packing unit of [1] is 8x8 pixel matrix, and this size will cause access redundancy for 4x4 block based motion compensation. Another kind of algorithms is DPCM-based [2]. By taking the intra prediction information of H.264 encoder to remove the spatial redundancy and combining Golomb-Rice coding, DPCM-based method achieves good quality. However, this method needs iteration several times to get the most appropriate quantization level. Thus compression cycle for each coding unit is not a constant, and leads to the complicate embedded compressor design. In [4], the authors modified Hadamard transform and combined with Golomb-Rice coding. With the shortest encoded cycles, MHT becomes the most flexible embedded compression scheme to be embedded into a video decoder.

In this paper, a new transform-based lossy embedded compression scheme is proposed. Taking 4x4 pixels as a coding unit and each unit is compressed with compression ratio two, only 64 bits is needed to store in external memory. And with the simple modified bit plane zonal coding, the decoding process of a 4x4 block can be done within 4 cycles including the data fetching. This algorithm can be pipelined into two stages. A MB only needs 34 cycles to decode and has 5.29dB quality improvement compared with MHT [4].

The rest of the paper is organized as follows. The proposed algorithm is introduced in section 2. The hardware implementation is introduced in section 3. In section 4, the detail of integration with H.264 decoder is presented and section 5 shows the evaluation result. The conclusions of this paper are presented in section 6.

II. PROPOSED EMBEDDED COMPRESSION ALGORITHM

A. Algorithm

The compression is conducted on a 4x4 pixel matrix (128 bits) obtained from the output of deblocking filter, and the

* Work supported by the NSC of Taiwan, R.O.C under contract NSC96-2221-009-180.

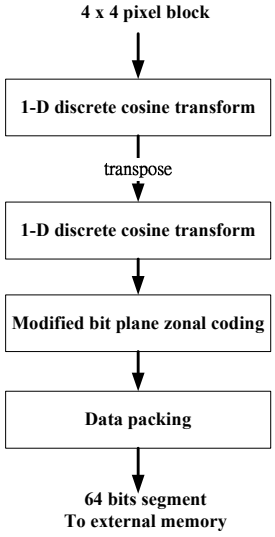

Fig. 1. Flowchart of the proposed embedded compression algorithm

compression ratio is fixed at two. Each 4x4 unit will become a 64 bits package after compression. Fixed compression ratio leads to constant amount of the coded data, so this EC scheme ensures the ability of random access without extra memory to record the segment address of coded data. Moreover, the 4x4 block unit is the basic coding unit in H.264 standard, and makes the data access in an efficient way.

Fig. 1 shows flowchart of the proposed algorithm. The overall compression scheme is formed by two parts: 1) two dimensions discrete cosine transform (DCT) and 2) modified bit plane zonal coding (MBPZ). Two dimensions DCT is composed by two one dimension, 4 points DCT. The discrete cosine transforms is a technique for converting a signal into elementary frequency components and it is widely used in image compression. For human visual system, human eyes are more sensitive on low frequency component of a picture and less sensitive on high frequency component. The DCT can gather the relative important low frequency signal on up left corner, and the most high frequency in down right corner. Thus the DCT combines with bit plane zonal coding with original point at up left corner can efficiently collect the information.

The second part is to perform modified bit plane zonal coding on 16 coefficients output from DCT. First, we reverse those negative coefficients into positive value and mark a "1" at the same position of sign bit plane. Sign bit plane can be considered as union of sign flags for each coefficient. Second, to improve the coding efficiency, we record the start plane. Search each bit planes from MSB to LSB (not including sign bit plane), and the first plane contains nonzero bits is the start plane. To avoid adding too much extra cycles and to simplify the hardware complexity in system view, we use one simple type only for recording each bit plane, though other complex types which can be more scrimping on bit using do exist. For each bit plane, we simply record the maximum row (RMAX) and column (CMAX) which have a "1" in this row or column, and then pack the bits in this zone which is enclosed by RMAX and CMAX. 4 bits are used to record RMAX/CMAX of each plane. And then, we packed the sign bit plane. Since we have only 64 bits budget for each 4x4 unit, the situation of unable to pack all the information may be happened frequently. Since not every coefficient can be packed, packing whole sign bit plane may become a waste. So we take the maximum value of RMAX and CMAX out of each packed bit plane and packing useful sign bits only by using those two boundaries. Under this method we will not waste extra bits to pack those unused sign bits, and the RMAX/CMAX of sign bit plane need not to be packed since they can be derived from the previous coded information. Finally, the end plane needs to be estimated and packed to specify when to stop.

B. Packing Mechanism

The overall packing scheme is introduced in this part. After doing discrete cosine transform, we get 16 coefficients from each 4x4 block. There are 15 AC coefficients and one DC coefficient. Sine DC coefficient is the average value and is the most important in transform, we reserve 8 bits budget for the DC coefficient of every 4 x 4 block. DC coefficient is always positive, so we don't have to worry about the sign bit for DC coefficient. For the rest 15 coefficients, we first packed the start plane and end plane (6 bits total). Then, we separate RMAX/CMAX and plane content of those planes which are between start plane and end plane, and connecting all the RMAX/CMAX together and all plane content together respectively. Sign bit information is inserted between the CMAX/RMAX and the plane content. Fig .2 shows the compressed segment format.

III. HARDWARE DESIGN

A. Discrete Cosine Transform

The hardware design of DCT is referred from Lee's architecture [5]. This architecture can maintain the same performance with original DCT while reduced the number of multiplications to about half of those required by the existing efficient algorithms. This design allows us to take the advantage of DCT while not suffering from its hardware complexity.

B. Modified Bit Plane Zonal Coding and Data Packing

There is a combinational block dealing with coefficients to derive the RMAX/CMAX and plane content of each plane. To serialize the plane information in one cycle, we propose the content adaptive ripple architecture to solve the problem.

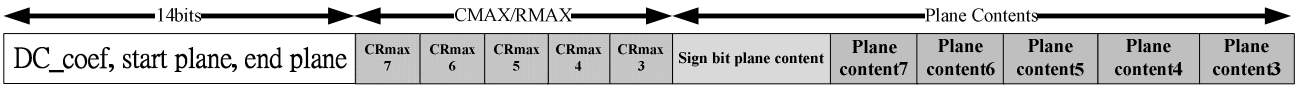

Fig. 2. Compressed 64 bits data format

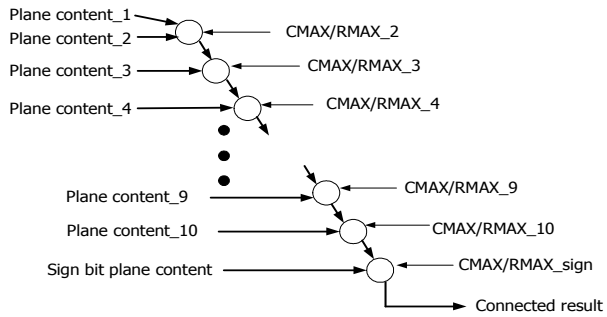

Fig. 3. Context adaptive ripple architecture

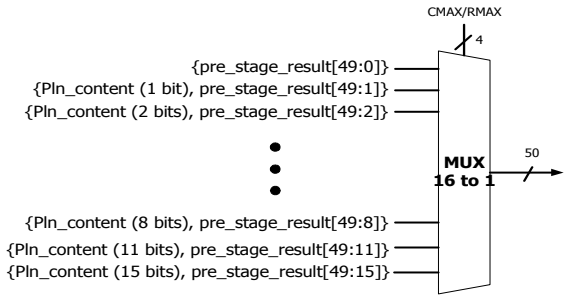

Fig. 4. The MUX structure

The basic concept is shown in Fig. 3. The 11 lines at left represent the 10 plane contents pulsing 1 sign bit plane content. Each circle represents a 16 to 1 MUX controlled by 4 bits RMAX/CMAX and is shown in Fig. 4. By the ripple behavior, the wire at the end of the flow is the connected result. Notice that we embedded this compressor into our 108MHz decoder, thus one cycle time is enough to finish our ripple processing.

C. Encoder Design and Decoder Design

Fig. 5 shows the pipeline architecture of compressor design. Since compressor has more time to handle the encoding process, we use three stages here and each stage needs 4 cycles. This design with longer cycles can shrink the gate count by reducing one dimension four points DCT units. Under this design, a MB needs 72 cycles to encode.

Fig. 6 shows the architecture of decompressor. To provide data for motion compensation unit suitably, the decompressor must support higher throughput inevitably. The decompressor is divided into two stages and each stage needs 2 cycles, a 4x4 block needs 4 cycles to decode, including the data-fetching. Decoding a MB just needs 34 cycles.

IV. INTEGRATION WITH H.264 DECODER

Fig. 7 shows the block diagram of the H.264 video decoder containing an embedded compressor. Embedded compressor works as the interface between decoder IP and external memory. And the EC mechanism is ready to be used by adding extra address control logic. Since the compression ratio of our embedded compressor is two, each data is compressed into half of the original size. The address control logic is very easy to implement.

The H.264 decoder works at 108 MHz, performing the HD1080i@30fps. The EC compress the data from deblocking filter, and 4x4 blocks will become 64 bits

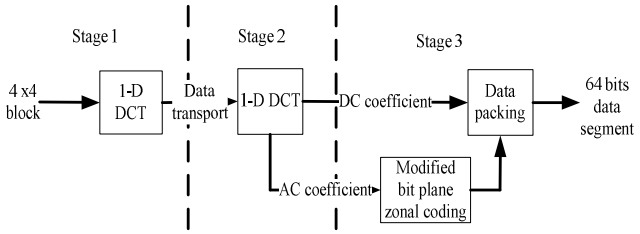

Fig. 5. Encoder architecture

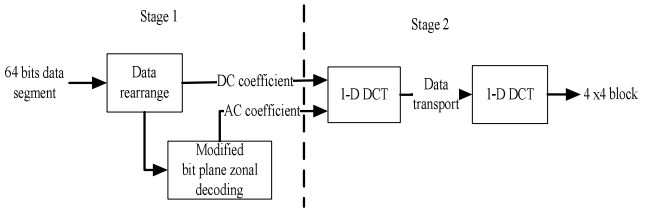

Fig. 6. Decoder architecture

segments and then stored into off-chip memory. Thus, the data access times of off-chip memory are half of the original access times for the system without an embedded compressor. The embedded decompressor decompresses data from external memory and sends them to motion compensation unit. The system bus bandwidth is default as 32 bits and the external memory is 32 bits per entry, so the original system takes 4 pixels as accessing data unit. The access behavior of motion compensation with/without embedded compressor can be analyzed as follows. If the requested 4x4 blocks are perfectly aligned with the coded 4x4 blocks, only 2 cycles are needed to fetch the 4x4 block while the original system needs 4 cycles to fetch. For the needed block not aligned with the coded blocks in only one direction, the system with embedded compressor needs to decode two blocks to derive the needed data, so 4 cycles are needed. The original system, taking 4 pixels as accessing unit, needs 4 cycles to fetch data. For the needed 4x4 block not aligned with the coded data in both vertical and horizontal directions, the system with EC needs decoding four 4x4 blocks and 8 cycles are needed for data fetching. The original system needs 8 cycles too. For the final situation, the sub pixel case, a 4x4 block needs a 9x9 pixels block to finish the motion compensation. 18 cycles is needed for EC while original system needs 27 cycles. From the analysis above, we can see that H.264 decoder with an embedded compressor does reduce the access times and can efficiently reduce the access power consumption.

V. EVALUATION RESULT

Software implementation of the proposed algorithm is integrated with JM12.4. The reference frame is compressed by the proposed algorithm and compared with the result of previous work using MHT and GR coding respectively. The test sequences are Foreman, Stefan, Mobile and Akiyo in CIF format and Station in HDTV format. All sequences are organized in one I frame follows nine P frames format. For each sequence, 100 frames are used to compute the average PSNR value reference to the original sequences.

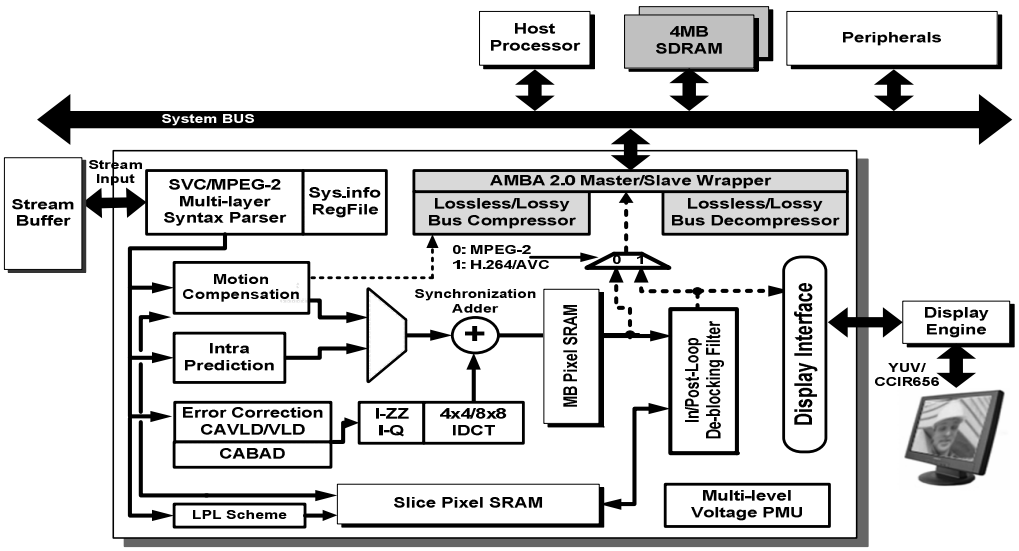

Fig. 7. H.264 decoder architecture with proposed embedded compression algorithm

Table. 1 is the simulation result of five sequences and shows the performance of original H.264 decoder without any recompression, embedded with MHT and embedded with our proposed algorithm.

Table 1. PSNR comparison

Test Sequence	Algorithm (@CR = 2.0)	PSNR (dB) (original)	PSNR lost (dB)
Forman (CIF)	original	35.57	0
	MHT	29.48	6.08
	proposed	34.21	1.36
Stefan (CIF)	original	36.02	0
	MHT	28.64	7.38
	proposed	33.86	2.16
Mobile (CIF)	original	33.75	0
	MHT	23.10	10.65
	proposed	29.27	4.48
Akiyo (CIF)	original	39.64	0
	MHT	32.17	7.47
	proposed	38.33	1.31
Station (HDTV)	original	38.85	0
	MHT	32.97	5.88
	proposed	37.13	1.72

The proposed method maintains better quality in slow motion sequences than high motion sequences. However, performance of the proposed method in all sequences is better than previous MHT work. The average PSNR degradation of proposed method is 2.21dB while the MHT is 7.5dB. Fig. 8 shows the embedded result with Station sequence in HDTV format. Although the quality drop is inevitable, the proposed method efficiently slows down the speed of decay than previous MHT work.

VI. CONCLUSIONS

In this paper, we proposed a flexible algorithm which achieves good coding efficiency and is suitable to be integrated with any video decoder. The proposed architecture is synthesized with 90-nm CMOS standard-cell library. The operation frequency is 108 MHz. The gate counts of proposed algorithm for compressor/decompressor are 15.8K/14.2K respectively. With the help of this embedded compression engine, we can reduce the bandwidth requirement and the external frame memory size. The proposed architecture costs 30K gate counts and deals with a 4x4 block unit while MHT costs 20K gate counts in dealing with a 1x8 pixels array. The proposed algorithm not only gains 5.29dB in picture quality but also achieves an area-efficient hardware implementation.

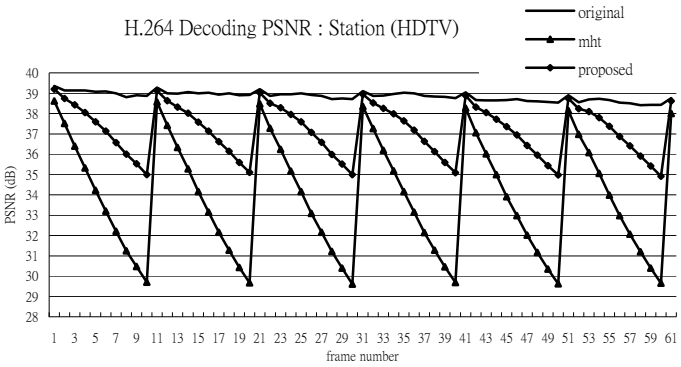

Fig. 8. Simulation result of Station sequence(HDTV)

ACKNOWLEDGMENT

The authors would like to thank Prof. T.-H. Chiang, Dr. T.-M. Liu and colleagues of the SI2 group of National Chiao Tung University for fruitful assistances on this work.

REFERENCES

[1] R. J van der Vleuten et al, "Low-complexity scalable DCT image compression", IEEE Proc. Image Processing, vol.3 pp. 837-840, Sep. 2000

[2] Yongje Lee; Chae-Eun Rhee; Hyuk-Jae Lee; "A New Frame Recompression Algorithm Integrated with H.264 Video Compression" Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium, pp. 1621-1624, May 2007

[3] A. Bourge and J. Jung, "Low-Power H.264 Video Decoder with Graceful Degradation," SPIE Proc. Visual Commun. and Image Process., vol. 5308, pp. 1234-1245, Jan. 2004

[4] T.-Y. Lee, "A New Frame-Recompression Algorithm and its Hardware Design for MPEG-2 Video Decoders," IEEE Trans. CSVT, vol. 13, no. 6, pp. 529-534, June 2003.

[5] Byeong Lee, "A new algorithm to compute the discrete cosine Transform" Acoustics, Speech, and Signal Processing, IEEE Transactions on Volume 32, Issue 6, pp:1243-1245, Dec 1984