Efficient Schemes for Bypass Flag Read and Write for 1-D Lossless Frame Memory Compression in the Hardware-based Video Encoder

Ji Hun Jang[1] and Chae Eun Rhee[2]
[1, 2] Department of Information and Communication, Inha University
Yong hyun-dong 253, Nam-gu, Incheon, Republic of Korea
E-mail: [1]jhjang@inha.edu, [2]chae.rhee@inha.ac.kr

Abstract: The frame memory compression (FMC) is one of solutions to mitigate the memory size and bandwidth burdens for the reference frames used in the inter-prediction. The lossless FMC keeps the image quality, whereas the compression ratio is not guaranteed. In the worst case, the required memory size with FMC is more than that without FMC. To manage this case, the encoding step can be bypassed and the uncompressed data are stored with bypass flags. Mostly, the overhead from bypass flags depends on the size of compression units. However, if the bypass flags are stored to the external memory and writing and reading are done through the system bus, the trade-off between internal memory space and bus bandwidth needs to be considered carefully. In this paper, the schemes for bypass flag read and write are proposed to reduce the bandwidth effectively with a reasonable internal memory overhead.

Keywords—**Bypass flag, Frame memory compression, Video codec, system bus, memory bandwidth**

1. Introduction

Most video coding standards adopt inter-prediction which exploits temporal redundancy between frames to improve coding efficiency where previous encoded frames should be stored for the reference. Accordingly, the memory access bandwidth to read and write reference frames also significantly increases [1]. The frame memory compression (FMC) can be one of solutions to mitigate the memory size and memory bandwidth burdens. The FMC encodes the reference frames before storing to the frame memory and thus, the memory size as well as the memory bandwidth is reduced. There are two types of the FMC, lossy and lossless approaches. The lossy FMC compresses the frame with the target compression ratio by discarding some data. Both memory size and bandwidth are reduced at the sacrifice of data loss. On the contrary, the lossless FMC compresses the frame without loss. The frame image quality is kept but the compression ratio is not guaranteed. In the worst case, the required memory size with FMC is more than that without FMC. For FMC, various compression algorithms are used depending on applications and system requirements. To avoid the additional memory usage, schemes to remove temporal redundancy are not used.

Set partitioning in hierarchical trees (SPIHT) [2] is one of the lossless compression schemes and is frequently used for FMC. No-list SPIHT (NLS) is the modified design for hardware implementation [3]. When the NLS is used for FMC, the compressed data size happens to be 56% larger than the uncompressed data size in some cases. To manage this case, the encoding step can be bypassed. In other words, not the compressed data but the uncompressed data are stored when the compressed data increases during NLS. Consequently, even in the worst case, the required memory size does not increase. In the decoding step of FMC, the decoder needs to distinguish the bypassed data from the compressed data. Thus, for every compression unit, 1 bit bypass flag is used to tell whether the current unit is compressed or not. The decoder accepts both data and a bypass flag. If the bypass flag=1, a decoding step is skipped. To apply this bypass option, the memory space for bypass flags is additionally required. However, considering the worst case such as 56% increase in data size, the memory overhead caused by bypass flags is small. The overhead from bypass flags is different depending on the compression unit. Table 1 shows the required memory size depending on whether or not the bypass scheme is used. Four frames of YUV420 videos with a resolution of 3840x2160 are encoded. The second row represents the memory size for the compressed data, whereas the third row represents the memory size for the bypass flag. Both assume the worst cases. Compared to the original NLS without a bypass scheme in the second column, the required memory size with a bypass scheme is much less even including the space for the bypass flag as shown from third to seventh columns. In the third and fourth columns, the compression unit is a 2-dimentional (2-D) block where the compression unit is relatively large. Thus, the additional space for bypass flags is quite marginal. However, in the case of 1-dimentional (1-D) compression from fifth to seventh column, the spaces for bypass flags are 48.6 Kbyte, 194.4 Kbyte and 777.6 Kbyte for 128, 32 and 8 pixel compression unit, respectively. Compared to 2D compression, the space for bypass flags is too large to store in the internal memory. If the bypass flags are stored to the external memory, writing and reading through the system bus are carefully designed considering the trade-off between the internal memory space and bandwidth overhead.

Table 1 Required Memory size

	Original NLS	Compression Unit size(pixels)				
		2D		1D		
		32x32	16x16	128x1	32x1	8x1
Compressed Data (Mbyte)	77.6	49.8	49.8	49.8	49.8	49.8
Bypass flag (Kbyte)	0	6.1	24.3	48.6	194.4	777.6

2. Proposed Bypass Flag Read and Write Schemes

In this paper, the system-level trade-off is analyzied to read and write bypass flags considering communication through bus, inernel memory size and bus bandwidth. Given the system constraints, the efficient read/write scheme for bypass-flags is proposed.

2. 1 System environment

Figure 1 shows the system environment assumed in this paper. The camera module accepts the video. YUV components of images are transmitted to the 1-D FMC encoder module in a raster-scan order via an AXI bus. In the 1-D FMC encoder, the image is encoded with a 32-pixel unit using SPIHT algorithm. The compressed data along with a bypass flag are written to the external memory through an AXI bus. Now, a codec module reads these input video to encode where 32x32 pixels data needs to be read since most video codecs encode data in a 2-D block-based manner. The 1-D FMC decoder checks the requests and the address from a codec and then reads the compressed data and bypass flags from the external memory. If the bypass flag is turned on, the decoding process is performed and the uncompressed data are transferred to the codec. Otherwise, the read data from the external memory are directly transferred to the codec without decoding. The AXI bus with a 128 bit data width is used and the random read and write of Y, U and V components are assumed.

In the codec system shown in Figure 1, system level factors need to be considered in addition to compression efficiency. In most cases, the camera module accepts the video in a raster scan order. Naturally, bypass flags are generated in an 1-D order while 1-D FMC encoding. If the codec reads the compressed data in an 1-D order, bypass flags stored in 1-D order can be read and used on the fly. However, most modern codecs perform a block-based coding to increase the compression efficiency and thus, bypass flags should be provided to an encoder in a 2-D order. It means that a mismatch occurs between memory access units of reading and writing. To cope with this mismatch, bypass flags stored in an 1-D order should be read from memory as much as height of the block which is used in the codec. If this multiple lines of bypass flags are stored temporarlly inside the FMC decoder, quite large internal memory is required. Otherwise, the FMC decoder needs to read bypass flags repeatedly and thus the bus bandwidth can be increased significantly.

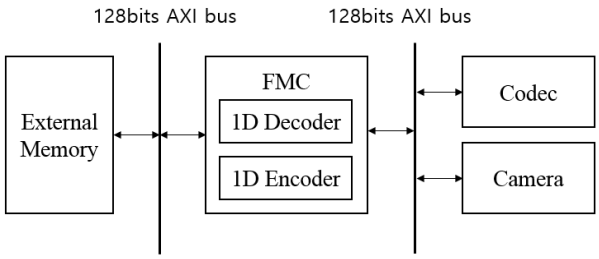

Figure 1. System environment

2. 2 Efficient Write Scheme

During FMC encoding, if 1 bit bypass flag for each compression unit is stored every time to the external memory through the bus with a 128 bit width, 127 bits among 128 bits are dummy bits. This is a significant waste of bus bandwidth. If 128 bypass flags which correspond to one pixel line of a 4k size frame are stored in the internal memory temporarily and then transferred to the external memory, the bus bandwidth is reduced. To do this, the 128 bit size internal memory is additionally required. If the random inputs of Y, U and V components should be supported like a system environment explained in Figure 1, 256(128+64+64) bit size internal memory is necessary. In other words, there is a trade-off between the internal memory size and the bus bandwidth. This trade-off is shown in Figure 2. It is assumed that the video size is 3840x2160 and its frame rate is 30 fps. The random input of YUV420 is supported. The horizontal axis represents the internal memory size, whereas the vertical axis represents the required bus bandwidth which depends on the internal memory size. For the bypass flag write, both input and output of the FMC encoder are 1-D and the encoding is performed on the fly. Thus, the required internal memory size is not so large.

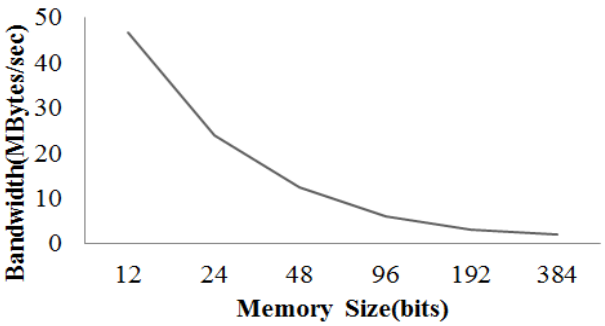

Figure 2. Trade off between Memory Size and Bandwidth in Write Process

2. 3 Efficient Read Scheme

For the bypass flag read, the codec wants to read data with a 32x32 block unit. For a 32x32 block decoding, the bypass flag for a 32-pixel unit should be read 32 times. Considering Y, U and V components, 64-time read access are required for each 32x32 block. To save the bus bandwidth, the bypass flags corresponding to 32 pixel lines of a frame can be stored in the internal memory in the similar way with the proposed bypass flag write scheme. For a YUV420 format, the bypass flags for 64 (32+16+16) pixel lines need to be stored. One pixel line of 4k size frames with a 32 compression unit needs 128 bypass flags. Thus, the required internal memory size comes to 8,192 (128 x 64) bits. Here, the trade-off occurs between the internal memory size and the bus bandwidth as shown in Figure 3. In Figure 3, the horizontal axis represents the internal memory size, whereas the vertical axis represents the required bus bandwidth. The 3840x2160 resolution, 30 fps and YUV420 format are assumed. If 8192 bit size internal memory is used, the bandwidth is just 2.07 Mbyte/s.

If the 256 bit size internal memory (4x64) is used, only 4 bits of 128 bit bus width are stored and other 124 bits are discarded when one read access. After using 4 bit bypass flags which covers 128 pixels corresponding to four compression units, another memory accesses are required for bypass flags which are read and discarded before. In this case, the internal memory size is small but the required bandwidth is 46.66Mbyte/s.

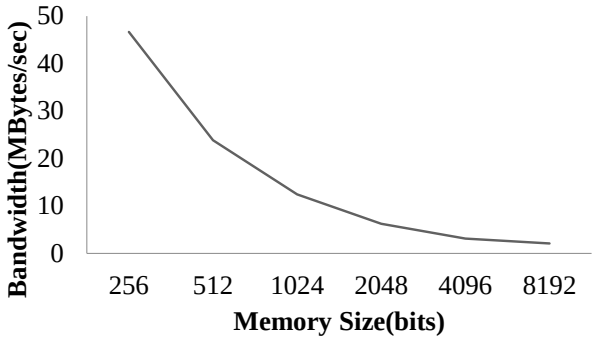

Figure 3. Trade off between Memory size and Bandwidth in Read Process

3. Experimental results

In this paper, considering the trade-off between the internal memory size and the bus bandwidth, 256 bits size and 2048 bits size internal memories are chosen for bypass flag write and read accesses. For experiments, Discrete Wavelet Transform (DWT) and SPIHT are used for FMC. As shown in Figure 4, DWT module and SPIHT module are in FMC encoder, whereas Inverse SPIHT (ISPIHT) module and inverse DWT (IDWT) module are in FMC decoder. After DWT transforms input data into a wavelet image, the wavelet image is compressed by SPIHT. Decoding process is reverse of the encoding process [4]. The compression unit of SPIHT is 32x1 pixels and DWT is performed by 2 levels.

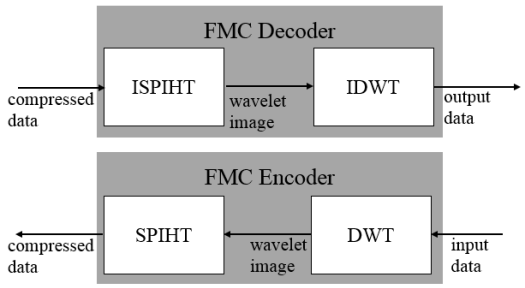

Figure 4. Architecture of FMC encoder/decoder

Table 2 shows the bandwidth required in system environment shown in Figure 1. The first and second columns represent resolutions and used video sequences, respectively. The third column represents the amount of compressed data from DWT+SPIHT encoding. It is just the output of the FMC encoder regardless of the assumed system environment. The fourth column denotes the bandwidth for bypass flags obtained by the proposed scheme in Section 2 The fifth column shows a bandwidth overhead caused by bypass flags read and write. Since 128 bit width bus is used and a bypass flag is generated every 32x1 pixels, the bandwidth required for bypass flags becomes minimum at an image width of 4096 (32x128) pixels. In Table 2, the bandwidth overhead is smallest at an image width of 3840. The smaller the width is, the greater the bandwidth overhead gets due to a lot of dummy bits.

Table 2 Bandwidth overhead caused by bypass flag when the proposed scheme is used

Resolution	Video Sequence	Bandwidth (Mbytes/frame)		Bandwidth Overhead (%)
		Compressed Data	Bypass Flag	
3840x2160	PeopleOnStreet	7.45	0.28	3.8
3840x2048	Traffic	6.86	0.26	3.8
1920x1080	BQTerrace	2.25	0.09	4.0
1920x1080	Kimono1	1.79	0.09	5.0
1920x1080	ParkScene	1.98	0.09	4.5
1280x720	Johnny	0.79	0.06	7.6

Table 3 shows the bandwidth according to the bus width and a video resolution in the system environment of Figure 1. The first column denotes the resolution of test videos. 4k (3840x2160) and 2k (1920x1080) videos are used because the FMC is likely to be used for high resolution videos. The second column denotes the bus width where 256, 128 and 64 bits bus widths are used. The third and fourth columns represent the average amount of compressed data from FMC encoding and the bandwidth for bypass flags, respectively. The fifth column shows the average bandwidth overhead due to the bypass flags. In Table 3, only bus width has been changed, whereas the internal memory size of FMC modules is same. Since the size of memory is chosen for 128 bits bus width, the bandwidth overhead is quite large in case of 256 bit bus width. The bypass flag overheads are 7.9% and 17% at 4k and 2k resolutions, respectively. If the FMC module reads bypass flags for 2k videos through 256 bits bus, the overhead is large because almost 76% bandwidth is occupied by dummy bits. To avoid this high rate of dummy bits, in 256 bit bus system, it is necessary to store bypass flags as much as 256 bits by using additional internal memory. From this observation, it is apparent that the size of internal memory should be determined in consideration of both the video resolution and the system bus width.

Table 3 Bandwidth overhead according to the bus width and video resolution

Resolution	Bus width (bit)	Average bandwidth (Mbytes/frame)		Average bandwidth overhead (%)
		Compressed Data	Bypass Flag	
4k	256	7	0.55	7.9
	128		0.28	4.0
	64		0.15	2.1
2k	256	2	0.34	17.0
	128		0.09	4.5
	64		0.04	2.0

4. Conclusion

In this paper, efficient bypass flag write/read scheme is proposed by considering the system level environment and analyzing the trade-off of various factors such as a bus width, the size of access unit, and the size of additional memory. The bandwidth required to read/write compressed data can be very different from the actual amount of compressed data depending on the read/write schemes through a system bus. If the further research is conducted, it is expected to get the bandwidth closer to the ideal compression ratio of the FMC module.

Acknowledgement

This work was supported by the Ministry of Science, ICT and Future Planning, Korea, through the Information Technology Research Center under Grant IITP-2016-H8501-16-1005 supervised by the Institute for Information and Communications Technology Promotion, and by the Basic Science Research Program through the National Research Foundation of Korea within the Ministry of Science, ICT and Future Planning under Grant NRF-2015R1C1A1A02037625.

References

[1] Ma, Yanzhuo, and Lijuan Kang. "Adaptive Granularity selection in Reference Picture Memory Compression." (2015).

[2] Said, Amir, and William A. Pearlman. "A new, fast, and efficient image codec based on set partitioning in hierarchical trees." Circuits and Systems for Video Technology, IEEE Transactions on 6.3 (1996): 243-250.

[3] Wheeler, Frederick W., and William A. Pearlman. "SPIHT image compression without lists." Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on. Vol. 6. IEEE, 2000.

[4] Kim, Sunwoong, et al. "A High-Throughput Hardware Design of a One-Dimensional SPIHT Algorithm." (2015).