IMPLEMENTATION OF REAL-TIME AIRBORNE VIDEO TELEMETRY SYSTEM

Ju-Hun Nam, Byeong-Doo Choi, Sung-Jea Ko
Department of Electronics Engineering, Korea University
lifespring@dali.korea.ac.kr

Bok-Ki Kim, Woon-Moon Lee, Nam-Sik Lee
Danam Systems Corporation., Korea

Jea-Taeg Yu
Agency for Defense Development, Korea

ABSTRACT

In this paper, we present an efficient real-time implementation technique for Motion-JPEG2000 video compression and its reconstruction used for a real-time Airborne Video Telemetry System. we utilize Motion JPEG2000 and 256-channel PCM Encoder was used for source coding in the developed system. Especially, in multiplexing and demultiplexing PCM encoded data, we use the continuous bit-stream format of the PCM encoded data so that any de-commutator can use it directly, after demultiplexing. Experimental results show that our proposed technique is a practical and an efficient DSP solution.

KEYWORDS

JPEG2000, Video Compression, PCM Encoder

INTRODUCTION

The basic purpose of a telemetry system is to gather data in a remote unit and transport them to users or recorders located in a base station [1]. A remote unit may be a vehicle, aeroplane, and missile. Since the data gathered at a remote unit, such as temperature, strain, frequency and other physical data, has low entropy in general, a telemetry system does not require a wide bandwidth to transmit the sensor data. However, as the need of airborne visual information to requires wide bandwidth increases, an advanced airborne telemetry system consist of individual dedicated video and telemetry communication systems. The telemetry system with video transmission capability uses two different carrier frequencies, one for the narrowband telemetry data and the other for the wideband video data.

The telemetry system we have implemented, is efficient in aspect of the physical size and the radio frequency bandwidth. It was achieved by using of the Motion-JPEG2000 (MJP2) and the combining the video and the telemetry data. It is named the Airborne Video and Telemetry System (AVTS). The AVTS is composed of following subsystems: PCM (Pulse Code Modulation) encoder, signal conditioner, video compressor based on the MJP2 and data processor of video and telemetry. In this paper, we show an efficient real-time implementation technique for MJP2 video compression and a reconstructing structure from a de-multiplexed serial data in its original continuous form.

SYSTEM OVERVIEW

Fig.1 shows the block diagram of the developed Airborne Remote Unit. It is composed of following subsystems: MJP2 Compressor, Signal Conditioner, PCM encoder, AVTDM (Airborne Video and Telemetry Data Module), Transmitter, where AVTDM multiplexes MJP2 compressed bit-stream and 256-channel telemetry data. Fig.2 shows the concept of the Base Station. The received RF signal is recovered by a bit-Synchronizer. The Ground Video Telemetry Data Module (GVTDM) separates the compressed video bit-stream data and the telemetry data. The system uses Reed-Solomon code for an error control coding.

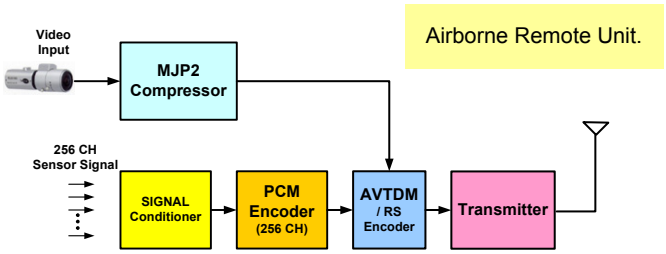

Fig.1. Airborne Remote Unit

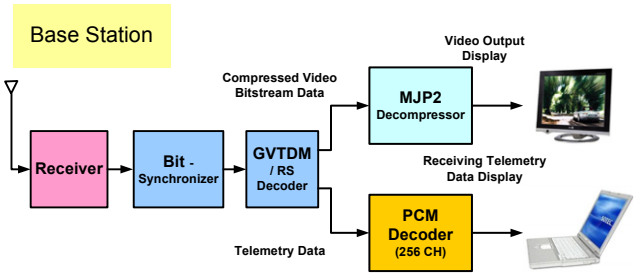

Fig.2. Base Station

VIDEO COMPRESSION

JPEG2000

JPEG2000 compression standard was created to provide high compression efficiency compared to JPEG [2]. It includes a rich set of features such as improved compression efficiency, lossy to lossless compression, and error resilience [3, 4]. MJP2 was intended to create a new coding system for video communication market and applications, based on JPEG2000 [5].

The fundamental building blocks of a typical JPEG2000 encoder are shown in Fig.3.

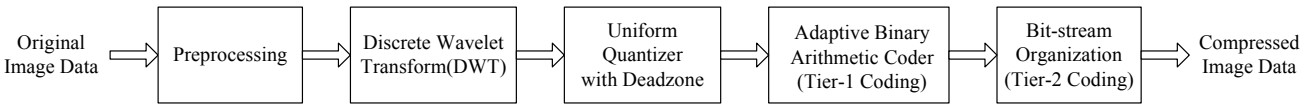

Fig. 3. JPEG2000 fundamental building blocks

The main demanding kernels of this standard are the Discrete Wavelet Transform (DWT) and the

Embedded Block Coding with Optimized Truncation (EBCOT) in the Tier-1. They require more than 85% of the encoding complexity and much more memory than those of the JPEG. Therefore it is very important to optimize these two modules in order to increase the performance. In this paper, to overcome these problems, we propose the computationally efficient DWT and EBCOT, which use SIMD instructions of the DSP and the overlapped block transferring (OBT).

OBT-Based Lifting Scheme for Efficient Cache Utilization

The lifting algorithm is known as a fast computing technique of the DWT. However, in view of the memory management, it still has severe cache-miss problem during the execution of the vertical wavelet filtering. A number of cache-misses make the processing time increase critically. Thus, even though the lifting algorithm requires a few execution of CPU, the processing time of DWT cannot be reduced remarkably without the memory management for reducing the number of cache-miss. Fig. 4 illustrates the reason why the cache-misses occur in the vertical filtering of DWT.

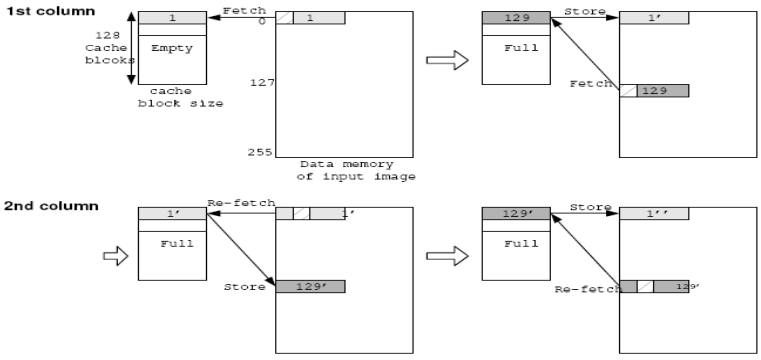

Fig.4. Cache-misses in vertical wavelet filtering

When filtering the first column, memory blocks are fetched as many as the height of an image. In the second column, memory blocks which were fetched in the filtering of the first column must be re-fetched because cache blocks of the upper-side data of an image are written to the external memory while down-side data is loaded in case that image height is larger than the number of cache blocks. These re-fetches of same data memory are iterated at every column.

In order to solve this problem, we developed an overlapped block transferring (OBT) method. This method is based on hierarchical memory architecture. The memory architecture of DSP is composed of two layers of data caches (L1D, L2D) and an external memory as shown in Fig. 5. The main feature of the OBT method is using of Direct Memory Access (DMA) operating independently without CPU execution for transferring the image data to L2D from the external

memory by block size equal to the cache size. Unlike the external memory, the address of the data memory on L2D is aligned by cache block size. Since the L2D could not hold a whole image of large size, DMA transfers data blocks from the external memory to L2D and from L2D to the external memory repeatedly by using double-buffering. Fig. 5 shows this mechanism. The data of the first column of blocks is transferred to L2D. After the first column of blocks is processed, this data is moved to the external memory and the next column of blocks is transferred to same location of L2D by DMA.

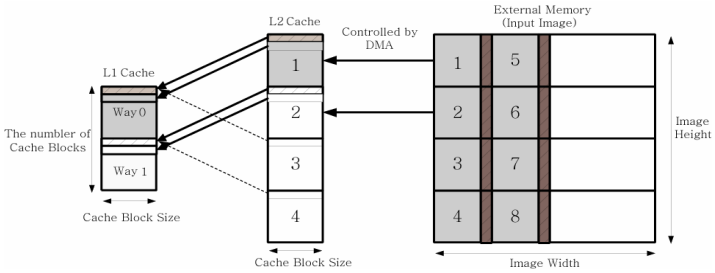

Fig.5. Memory manipulation of proposed OBT

Fig. 6 shows that the adjacent blocks are overlapped with each other along the horizontal direction. Area 1 in light gray is completely wavelet processed, whereas Area 2 in dark gray contains data lifted partially. Thus, the next block for the 2-D lifting includes Area 2 as well as Area 3 in light gray.

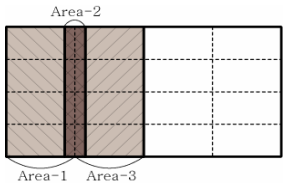

Fig.6. Overlapped block configuration

After remaining horizontal lifting steps for pixel values in Area 2 are completed, the 2-D lifting scheme is processed for Area 3. As a result, the data in light gray is fetched onto the cache one time, and the data in dark gray is fetched two times. It means that the cache miss rate is reduced drastically.

Optimization of the Lifting Algorithm Using SIMD Instructions and super scalar pipeline structure

Although the lifting algorithm reduces computational complexity as well as memory to be used, the algorithm still has much of computational burdens. To improve the performance of the lifting

algorithm we implement the lifting algorithm with SIMD instructions of DSP.

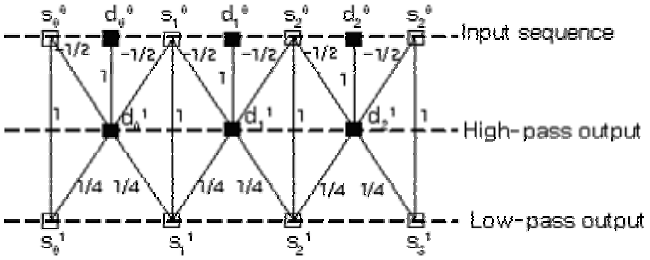

Fig.7. Lifting prediction/update steps for the (5,3) filterbank

The lifting operation consists of several steps. The basic idea is to first compute a trivial wavelet transform, also referred to as the lazy wavelet transform, by splitting the original 1-D signal into odd and even indexed subsequences and then modifying these values using alternating prediction and updating steps. Fig. 7 depicts an example of lifting steps corresponding to the integer (5,3) filter-bank. The sequences $\{d_i^0\}$ and $\{s_i^0\}$ denote the even and odd sequences, respectively, resulting from the application of the lazy wavelet transform to the input sequence. The prediction and updating steps are given as

$$d_i^1 = d_i^0 - \frac{1}{2}(s_i^0 + s_{i+1}^0) \tag{1}$$

$$s_i^1 = s_i^0 + \frac{1}{4}(d_{i-1}^1 + d_i^1) \tag{2}$$

Due to the simple structure of the (5,3) filter-bank, the output, $\{d_i^1\}$ and $\{s_i^1\}$ are actually the high- and low-pass output of the DWT filter, respectively. These processes are sequential updates of a value with a weighted sum of the adjacent values.

Fig. 8 shows the flowchart of the basic operation used for every pixel values in the lifting scheme, where the lambda in Fig. 8 for (1) and (2) is -1/2 and 1/4, respectively. Thus, for an N×M image, a generic C code requires 2×N×M 16-bit additions, N×M 16-bit multiplications, and 3×N×M 16-bit data load. Since a 16-bit multiplication, a 16-bit addition, and a 16-bit data load take 4 cycles, 1 cycle, and 5 cycles for execution in general respectively, the generic C code requires total 21×N×M cycles for processing an N×M image.

For the SIMD instruction of DSP, ”LDDW” can load 64bits, that is, four 16-bit values are loaded simultaneously. In addition, DSP provides multiplication and addition instructions for two 16-bit values. Since two instructions can be used simultaneously using the super scalar pipeline

structure of DSP, four 16-bit values can be added or multiplied at the same time. Thus, the use of the SIMD instructions reduces execution time drastically. For the case described above, the identical operation optimized with SIMD requires only 10×N×M cycles for processing an N×M image.

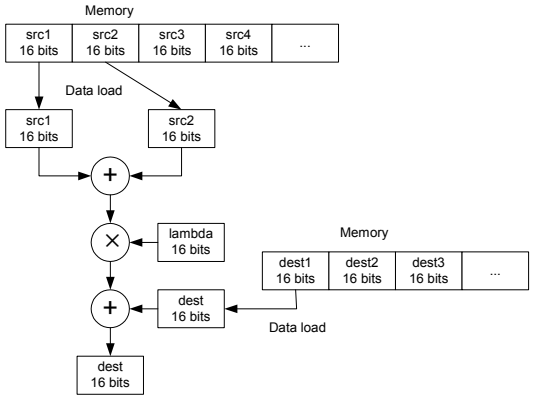

Fig.8. Basic operation used in the lifting scheme

RECONSTRUCTING STRUCTURE OF DE-MULTIPLEXED SERIAL DATA IN ITS ORIGINAL CONTINUOUS FORM

In a system we developed, PCM Encoded serial data is multiplexed and transmitted through the air, received and bit-synchronized, de-multiplexed and reconstructed to its original serial data as though there are no in-between multiplexing, de-multiplexing processes.

We propose a structure, used in this system, to re-build a continuous bit stream from packet-transmitted data.

Configuration

There are two systems, sender(S) and receiver(R). S multiplexes and transmits data, while R receives and de-multiplexes. Each system has its clock frequency f_S for sender S, f_R for receiver R, to output serial bit stream. And each frequency has its accuracy A_S, A_R.

In Fig. 9, f_S is a fixed logical value representing the serial data rate and A_S, A_R has fixed value according to the specific hardware, while f_R can have arbitrary value dynamically.

In these systems, in order for output serial data to be continuously streamed, the receiver frequency f_R must be within a reasonable boundary from the nominal value of the sender frequency f_S and the average of f_R's should converge to f_S.

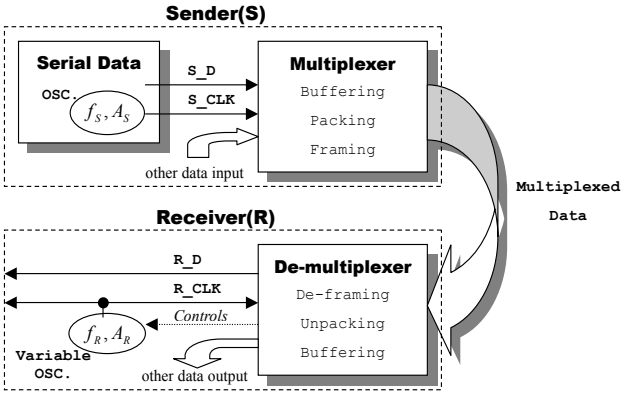

Fig.9. Systems configuration

Reconstructing Architecture

We wanted the controlling mechanism simple and stable, so decided to control f_R to have two frequency values between which original frequency f_S resides. If we call the two frequencies f_{RL} (lower) and f_{RH} (higher), we could describe their relations to f_S including accuracy values A_S, A_R as follows.

$$f_{RL}(1+A_R) \quad < \quad f_S(1 \pm A_S) \quad < \quad f_{RH}(1-A_R) \qquad (3)$$

Choose f_{RL} and f_{RH} so that the inequalities (3) could hold for any hardware specifics (A_S, A_R. e.g. max(A_S)<100ppm(0.01%)). Then control f_R to have f_{RL} or f_{RH} monitoring its fastness or slowness. In order to monitor f_R's fastness or slowness, we used the output buffer's data count. If the buffer is almost empty then f_R is faster than f_S, otherwise slower.

With this state information, we constructed a state machine to control f_R. In this way, the buffer's data count bounces approximately between "almost full" and "almost empty" periodically.

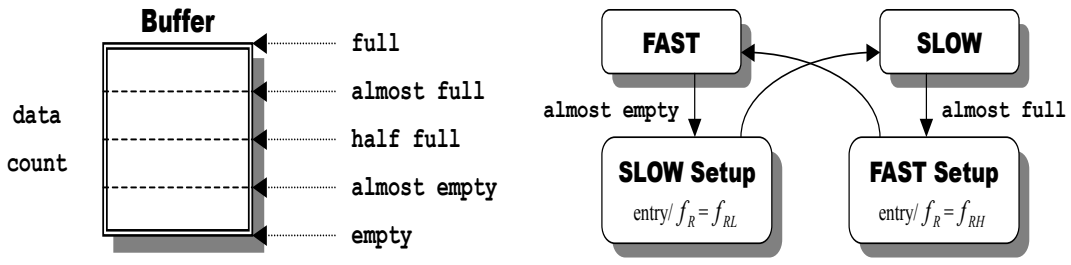

Fig.10. Buffer state information and control state machine

EXPERIMENTAL RESULTS

In the previous section, we proposed the OBT-based lifting scheme to increase the cache hit rate and the use of SIMD instructions to reduce the execution time of DWT. In this section, we demonstrate the performance improvement of the proposed method. Table 1 shows the number of cache-misses produced by the proposed method. As shown in Table 1, in the horizontal filtering, the proposed method produces some cache-misses due to the block overlapping. However, in the vertical filtering, the proposed method perfectly removes the cache-miss. Consequently, the proposed method reduces the cache-miss rate by 98%. Table 2 shows the processing time of DWT using the two existing method and the proposed method. There is no improvement in the horizontal filtering. But the proposed method tremendously reduces the processing time by 98% in the vertical filtering. Table 3 shows the performance improvement by using SIMD instructions of DSP. Since both the horizontal and the vertical lifting are applied for every pixel, DSP SIMD implementation can achieve 3.35 times faster computation time then the other.

Table 1. Comparison of the number of cache-misses

Image size	Lifting direction	Number of cache-misses	
		Conventional method	The proposed method
512×512	Horizontal	4096	4608
	Vertical	262114	0

Table 2. Comparison of the processing time of wavelet lifting scheme (512×512 image size)

Different method		Execution time of DWT			
		Horizontal(ms)	Vertical(ms)	Total(ms)	Speed-up
Original wavelet-lifting		12.74	659.35	672.09	1
Meerwald's method [5]	Row extension	12.98	143.77	156.75	4.28
	Aggregation	12.85	77.15	89.10	7.54
	Combination	13.02	61.27	74.29	9.04
Chatterjee's method [6]	Strip-mining	12.89	175.90	188.79	3.56
	Data layout	12.95	225.71	238.66	2.82
	Combination	12.87	115.40	128.27	5.24
Overlapped block transferring		17.94	17.35	35.29	19.04

Table 3. Comparison of the execution time of DWT between the conventional C code and the SIMD implementation

Lifting direction	Conventional C code	SIMD	Improvement
Horizontal	13.83 ms	4.22 ms	3.28
Vertical	13.35 ms	3.90 ms	3.42

CONCLUSIONS

In this paper we have proposed two efficient methods for MJP2, which are the OBT-based lifting scheme and a parallel processing method using SIMD instructions of DSP. We also proposed reconstructing structure of de-multiplexed serial data. By utilizing proposed methods, MJP2 encoding implementation meets common requirement for real-time video coding and we can use these system to process de-multiplexed bit stream data using the existing receiver systems without modifying them, just by connecting the serial output of the de-multiplexer to a de-commutator of the receiver systems.

REFERENCES

[1] Horan, S., Introduction to PCM Telemetering Systems, 2nd Edition, CRC PRESS, 2002

[2] Rabbani, M., Joshi, R. An overview of the JPEG2000 still image compression standard, Signal Processing:Image Communication, vol. 17, no. 1, January, 2002, pp.3-48

[3] Information Technology – JPEG2000 Image coding system:Part 1. ISO/IEC Internation Standard. 15444-1, 2000

[4] Taubman, D., High performance scalable image compression with EBCOT, IEEE Trans. Image Processing, vol. 9, 2000, pp.1158-1170

[5] Meerwald, P., Norecn, R., Uhl, A. Cache issues with JPEG2000 wavelet lifting, Proc. SPIE, Visual Communications and Image Processing, vol. 4671, 2002, pp.626-634

[6] Chatterjee, S., Brooks, C. D., Cache-efficient wavelet lifting in JPEG2000, IEEE International Conference on Multimedia and Expo., vol. 1, 2002, pp.797-800