# An Inter-frame/Inter-view Cache Architecture Design for Multi-view Video Decoders

Jui-Sheng Lee[1], Sheng-Han Wang[2], Chih-Tai Chou[2], Cheng-An Chien[2],
Hsiu-Cheng Chang[2] and Jiun-In Guo[1]
[1]Dept. of Electronics Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan
E-mail: {jslee.ee00g, jiguo}@nctu.edu.tw Tel: +886-3-5712121
[2]Dept. of Computer Science and Information Engineering, National Chung-Cheng University, Chia-Yi, Taiwan
E-mail: {wsh99m, cct99m, cca95m, changhsc}@cs.ccu.edu.tw Tel: +886-5-2720411

***Abstract*—In this paper we propose a low-bandwidth two-level inter-frame/inter-view cache architecture for a view scalable multi-view video decoder, which adopts two decoder cores to decode multi-view videos in parallel. The first level L1 cache is developed for the single video decoder core, which is able to reduce 60% bandwidth in doing inter-frame prediction in average. Moreover, we develop the second level L2 cache architecture to reuse the same reference data for doing inter-view prediction among different decoder cores, which can further reduce 35% bandwidth. By adopting the proposed two-level cache architecture for doing inter-frame/inter-view prediction, we can reduce 80% bandwidth through a view scalable multi-view video decoder implementation, which achieves real-time HD1080 dual-view video decoding.**

## I. INTRODUCTION

Nowadays, people require high quality video/image service in a more eager way. For example, one movie or one ball game could not be filmed by only one camera. Many cameras would be used simultaneously for meeting the demands of different view angles of viewers. To deliver this service, there are much more video data needed to be compressed. Although H.264/AVC is generally acknowledged the best video coding standard, if we apply the past video coding concepts, we need to process multiples of quantities of video data. As a result, the ITU-T formulates the Multi-view Video Coding (MVC) Standard which bases on H.264/AVC and takes advantage of video data dependency from different views to reduce the quantity of data to be processed as shown in Fig.1. Fig.2 shows the extended function which is called disparity estimation/compensation (DE/DC). Even support single view video decoding in real-time, there are some design challenges to be resolved, a fortiori to do real-time decoding different views in parallelism of multi-view video. In multi-view video decoding, motion/disparity compensation is one of major bottlenecks. It takes up more than 70% memory bandwidth in motion/disparity compensation for each view. Therefore, for the purpose of supporting DE/DC, the memory bandwidth is multiplied by the H.264/AVC.

Multi-view video is an upcoming multimedia application, for example, free-view point TV, 3D stereo TV, and so on. These services need higher performance for decoding different views at the same time. With regard to video coding bandwidth reduction, there are some researches having been proposed in literatures [1-4]. The Ref. [1] proposed a 2-D data mapping two-way cache consisting of 64 banks for SVC/MVC decoders. The Ref. [2] used a 6-way cache for data mapping to increase data reusability. The Ref. [3] proposed a data loading mechanism for promoting data reusability and reducing bandwidth. Some of the previous works only support single view decoding with H.264/AVC and did not propose any preload and update mechanism for their cache design.

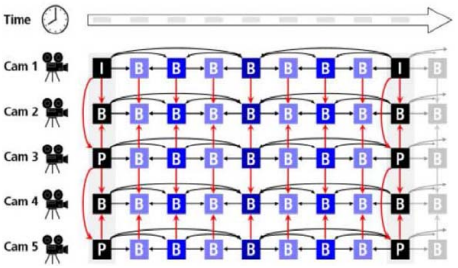


Fig.1 Multi-view video coding flow

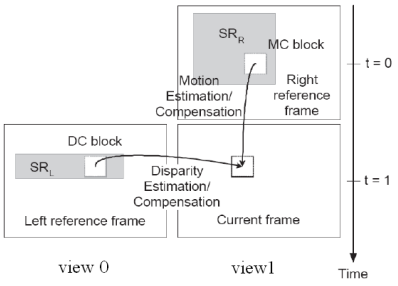


Fig.2 Inter-frame/inter-view prediction in MVC

For both of H.264/AVC and MVC, 6-tap filter is adopted for supporting sub-pixel interpolation in MC. According to 6-tap filter, there are some overlapped regions between adjacent blocks, as shown in Fig.3. Thus, we propose a level 1 cache for each decoder core in the view-scalable multi-core decoder architecture for multi-view video decoding. In every kind of cache design, there must be a tradeoff between cache size and data hit rate. Actually, it also encounters some problems of memory access latency and update mechanism in realistic system applications. To solve these problems we propose an

update mechanism according to the current decoding macroblock (MB) type. If the sub-block size is smaller (e.g., mode skip, 8x8, 8x4, 4x8), it tends to update entire cache data. But if the sub-block size is bigger (e.g., 16x16, 16x8, 8x16), it will choose to update partial cache data for reducing memory bandwidth.

In the two-level cache architecture based view scalable multi-view decoder system, we are able to decode different views of multi-view video simultaneously and support the view-scalable flexibility. With this feature, we could only adjust the number of MVC decoder cores and set the firmware parameters according to different system requirements.

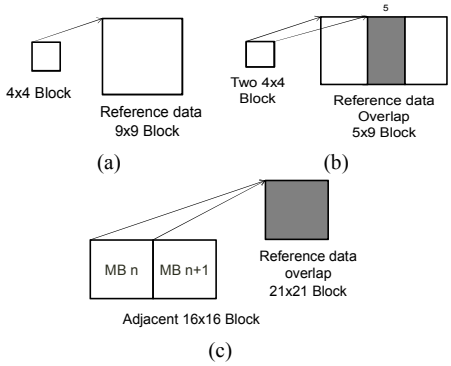


Fig.3 Example of data reuse in motion compensation

The rest of this paper is organized as follows. We describe the proposed inter-frame/inter-view cache architecture design in Section 2. The implementation result of the proposed design is shown in Section 3. Finally, we give a conclusion in Section 4.

## II. PROPOSED TWO-LEVEL CACHE ARCHITECTURE

For mitigating MC/DC bandwidth, a two-level cache architecture is proposed. In our proposed cache design, the real system application is regarded. For example, efficient bus and memory access mechanisms are involved in our design. Due to the H.264/AVC high profile and MVC standard, bi-direction reference (B-frame) is allowed. Therefore, the proposed cache is designed as two-way type for supporting forward (LIST0) and backward (LIST1) reference.

In the cache size analysis, first of all, we profile the bit-stream encoded by JMVC3.0. We found that the inter prediction in P-frame occupies about 54.67% (MB mode skip accounts for 10% totally), and more than 96% (MB mode skip accounts for 74% totally) in B-frame. It represents that MB mode skip often arises. Taking the group of picture (GOP) size 8 as an example, B-frame could account for more than 90%. It proves that how the proposed cache architecture can reduce more than 80% bandwidth by the good mechanism of data reusability. Although there is only supporting inter-frame and inter-view data cache in the proposed design.

### *A. Proposed inter-frame cache analysis*

We adopt JMVC 3.0 software for cache size and hit rate analysis. According to 6-tap filter for sub-pixel interpolation, the MC reference data would be (block width+5)*(block height+5) bytes. Consequently, the cache size must be (MB width+5)*(MB height+5) bytes at least. And the 32-bit and 64-bit bus width is often being employed. Thus, we primarily choose 32x32 bytes, 36x36 bytes, 40x40 bytes, 64x21 bytes, 64x24 bytes, and 64x28 bytes for the cache size analysis. As shown in Fig.4, it is obvious that the hit rate of cache size 40x40 bytes is the best choice in our experiment. In the following analysis, we will consider the condition of real system applications.

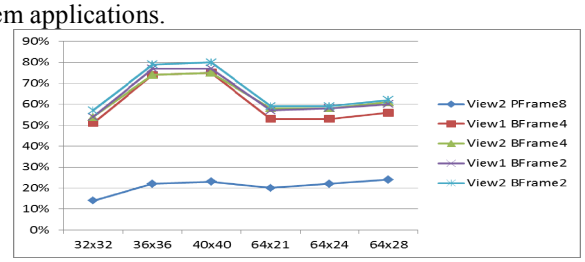


Fig.4 Inter-frame cache hit rate tendency on different cache sizes

Taking realistic system applications into account, we must encounter the external memory access latency. It is well known that if MC access external memory is crossing two different rows, it need to resend the access commend through bus to the memory controller. There is no doubt that both of bus and memory controller must have delay time in responding every request. And the mainstream bus architecture supports four kinds of burst lengths (1, 4, 8, and 16), which indicates the number of data could be accessed continuously by issuing one commend. Therefore, the cache size 40x40 bytes may be unsuitable for real system applications due to the fact that we have to send 3 commends sequentially as burst length 8, 1, and 1 for accessing one row data in 32-bit bus, which increases the bandwidth as shown in Fig.5.

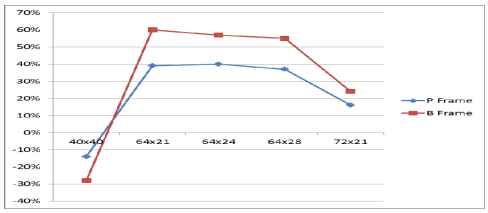


Fig.5 Average bandwidth reduction with different inter-frame cache sizes

Based on the above analysis and MB mode skip arises frequently in MVC, we can find out that the rectangle cache (64x21 bytes, 64x24 bytes, and 64x28 bytes) is much more appropriate in real system applications. Because the data row size of 64 only needs one commend (burst length 16) for data access and the frequent appearance of MB mode skip often needs to reference the neighboring MB data in horizontal way, it turns out that the rectangle cache is better. As shown in Fig.5, the cache size 64x21 bytes would have the best performance in bandwidth reduction.

In the proposed design, we also propose an efficient cache update mechanism on the basis of the current decoding MB type. If the sub-block size is smaller, we tend to update entire cache data, but if the sub-block size is bigger, we choose to

update partial cache data for reducing memory bandwidth. As we mentioned above, the MB mode skip often continuously appears in MVC bit-stream, and decoder would divide the current decoding MB into four 8x8 sub-blocks for doing MC. As shown in Fig.6, it assumes that the current decoding MB is mode skip while the reference data in cache is partially hit as the block H. Encountering this situation, the proposed mechanism would update entire cache for the much probably appearance of MB mode skip in decoding next MB. If we only update the missed data as block M, we should send two commends while the data is crossing two memory rows and suffer twice access latencies. By the proposed mechanism, we can increase cache hit rate of 8-10%.

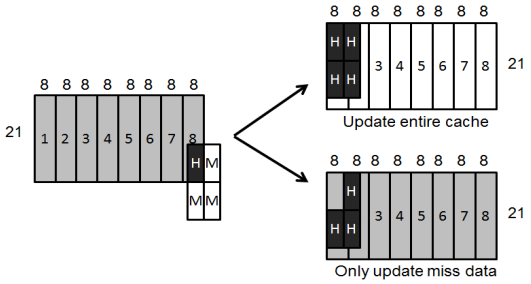


Fig.6 Inter-frame cache update mechanism in MB mode skip

As shown in Fig.7, the inter-frame cache is built inside every core of the proposed view-scalable MVC decoder system. Especially to deserve to be mentioned, the proposed system could be set to decode multi-view video in one decoder core by firmware, and the proposed inter-frame cache controller could also be set to support inter-view and inter-frame reference data access simultaneously.

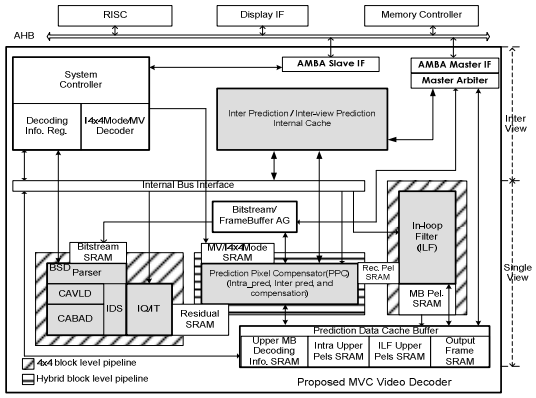


Fig.7 One core of view-scalable MVC decoder system

## B. *Proposed inter-view cache analysis*

As shown in Fig.8, there are many overlapped reference data in MVC. In order to increase the data reusability, the proposed inter-view cache is adopted between every core of view scalable MVC decoder for inter-view reference.

We analyze cache size and the associated hit rate by JMVC 3.0 software. We record the overlapped region in every frame which would be referenced simultaneously by two different views in disparity compensation to estimate the corresponding hit rate of each cache size. As shown in Fig.9, the cache size 64x64 bytes has the highest hit rate. And we take the external memory access latency into account, which could find that the rectangle cache is more efficient in real system applications. Although the width of cache size 64x64 bytes could access a row data by using one commend, but it costs more and only promotes about 8% in hit rate. Finally, we conclude the best choice is cache size 64x24 bytes for the MVC decoder.

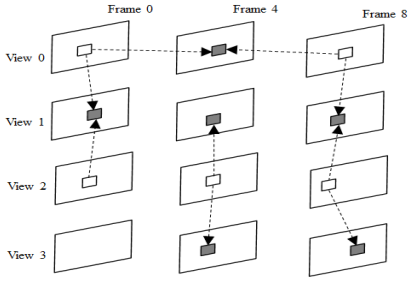


Fig.8 Inter-view reference in MVC

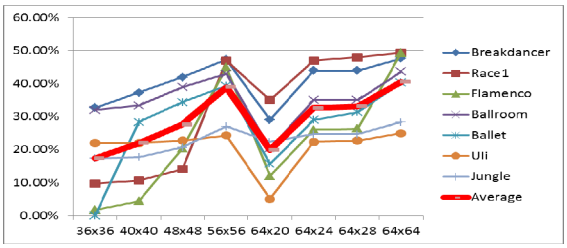


Fig.9 Inter-view cache hit rate tendency on different cache sizes

## III. IMPLEMENTATION RESULTS AND PERFORMANCE EVALUATION

We implement the cache into a pipeline architecture for increasing performance. As shown in Fig.10, after we reschedule the data preload, the decoder system could not stall until the data preload finished when it meets a cache miss.

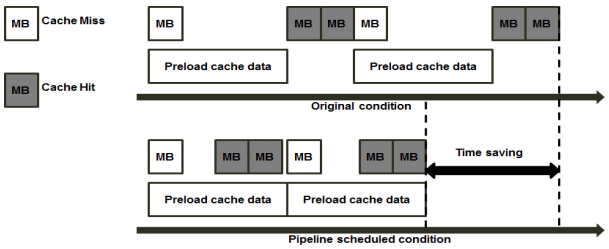


Fig.10 Example of pipeline schedule cache data preloading

Fig.11 shows that the inter-frame cache could reduce about 46% external access bandwidth at video resolution HD720 with access latency as 3 cycles. And when the access latency is increased to 30 cycles, our cache design actually could reduce more than 60% bandwidth at HD720. As shown in Fig.12, our inter-view cache could reduce about 26% bandwidth at video resolution XGA (1024x768) with external access latency as 3 cycles. And when the external latency is

increased to 30 cycles in some typical SoC, we even can achieve more than 40% bandwidth reduction.

These results show that both of our inter-frame and inter-view cache would have higher performance when encountering the higher external latency even if the video resolution is also increased.

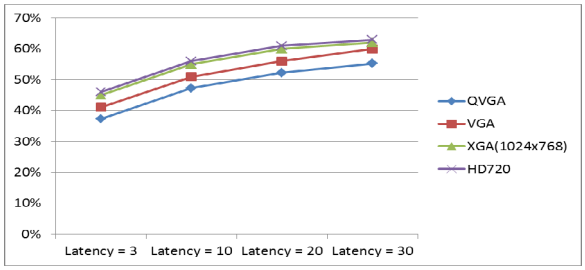


Fig.11 Inter-frame cache bandwidth reduction tendency

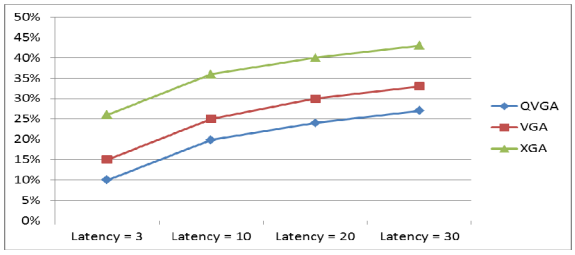


Fig.12 Inter-view cache bandwidth reduction tendency

As shown in Fig.13, these results are average bandwidth reduction with decoding video resolution VGA at 30fps. Here we separate the data loading bandwidth and external access latency for further comparison. As the second column in Fig.13, we can find out the external access latency would be reduced more than 70% by only applying our inter-frame two-way cache. It proves that our analysis on cache size and the proposed update mechanism is good for accessing external memory by efficiently sending access request.

As shown in table 1, the cache designs of Ref [1] and [3] are only support single view video decoding in H.264/AVC. And although the Ref [2] could support MVC decoding, but our MVC decoder system could support flexibility by L2 inter-view cache, which is not proposed by any previous works.

## IV. CONCLUSIONS

This paper presents a two-level cache architecture for H.264/AVC and view-scalable MVC decoder, we also propose an update mechanism for promoting performance. To support view-scalable for MVC, we further present inter-views cache architecture to utilize the overlap data in inter-view prediction. According to the simulation result the proposed cache architecture can achieve more than 70% bandwidth reduction in inter-frame cache and 30% bandwidth reduction in inter-view cache, the total bandwidth reduction can reach 80% as integrating inter-frame cache and inter-views cache architecture. The proposed system can achieve HD1080 dual-view video decoding in real-time.

TABLE I
PERFORMANCE COMPARISON WITH PREVIOUS WORKS

| Design Feature | [1] | [2] | [3] | Proposed Design |
|---|---|---|---|---|
| **Cache Architecture** | Single-MB cache | Two-way cache | Six-way cache | Two-way cache |
| **Support** | H264/AVC | H264/AVC MVC | H264/AVC | H264/AVC MVC |
| **Prediction Mode** | Inter | Inter Inter-view | Inter Intra | Inter Inter-view |
| **View Scalable** | N/A | N/A | N/A | 2, 3, 4 views |
| **Latency BW Reduction** | 16% | 30% | 45% | 77% |
| **Data BW Reduction** | 6% | 41% | 35% | 15% |
| **L2-Cache BW Reduction** | N/A | N/A | N/A | 30% |
| **Total BW Reduction** | 12% | 34% | 40% | 80% |

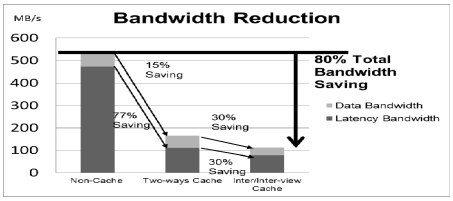


Fig.13 Bandwidth reduction of proposed cache architecture in VGA two-view IBBBBBBBP @30fps

## REFERENCES


[1] T.-D. Chuang et al., "A 59.5mW Scalable/Multi-view Video Decoder Chip for Quad/3D Full HDTV and Video Streaming Applications," in *Proc. ISSCC*, 2010, pp.330-331.

[2] T.-D. Chuang, L.-M. Chang, T.-W. Chiu, Y.-H. Chen, and L.-G. Chen, "Bandwidth-Efficient Cache-based Motion Compensation Architecture with DRAM-friendly Data Access Control," in *Proc. ICASSP*, 2009, pp. 2009-2012.

[3] Y. Li, Y.-M. Qu, and Y. He, "Memory Cache Based Motion Compensation Architecture for HDTV H.264/AVC Decoder," in *Proc. ISCAS*, pp.2906-2909.

[4] C.-Y. Tsai, T.-C. Chen, T.-W. Chen and L.-G. Chen, "Bandwidth optimized motion compensation hardware design for H.264/AVC HDTV decoder," in *Proc. ISCAS*, 2005, pp.273-276.

[5] Joint Video Team (JVT) reference software JMVC 3.0.

[6] 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.

[7] Y.-S. Ho, K.-J. Oh, "Overview of Multi-view Video Coding," in *Proc. IWSSIP*, 2007, pp.5–12.

[8] C.-A. Chien, H.-C. Chang, and J.-I. Guo, "A High Throughput In-Loop De-blocking Filter Supporting H.264/AVC BP/MP/HP Video Coding," in *Proc. APCCAS*, 2008, pp.312-315.

[9] Y.-C. Yang, and J.-I. Guo, "High-Throughput H.264/AVC High-Profile CABAC Decoder for HDTV Applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 9, Sep 2009.

[10] R.-S. Wang, and Y. Wang, "Multiview video sequence analysis, compression, and virtual viewpoint synthesis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 3, pp.397-410, Apr 2000.