ALGORITHM AND ARCHITECTURE DESIGN OF CACHE SYSTEM FOR MOTION ESTIMATION IN HIGH DEFINITION H.264/AVC

Wei-Yin Chen, Li-Fu Ding, Pei-Kuei Tsung, and Liang-Gee Chen

DSP/IC Design Lab, Graduate Institute of Electronics Engineering,
National Taiwan University, Taipei, Taiwan
Email: {cvictor, lifu, iceworm, lgchen}@video.ee.ntu.edu.tw

ABSTRACT

High Definition (HD) video compression enables vivid reproduction of scenes. However, Motion Estimation (ME) requires large memory capacity and huge memory bandwidth, which are undesirable in many platforms including ASIC and SoC. In this paper, an algorithm and architecture design of cache system and fast ME in HD H.264/AVC are proposed. With the proposed cache system and hardware-oriented fast ME algorithm, the rate-distortion performance is maintained within 0.03dB difference, the size of on-chip memory reduced to only 10% to 21% of original size, while the external memory bandwidth from cache refill is also 18% to 56% less than that of level C data reuse scheme with vertical ±64 search range.

Index Terms— Video coding, Cache memories, Motion analysis, Motion Estimation, H.264/AVC

1. INTRODUCTION

With the emerging need of high quality multimedia communication, Joint Video Team (JVT) proposed H.264/AVC [1] as the next generation video coding standard, which achieves very high coding efficiency. H.264/AVC can be used in enormous applications such as video conferences, HD/Blu-ray disks, camcorders, home theater, and so on so forth.

H.264/AVC introduces several useful features, by which the extraordinary coding efficiency could be achieved [2]. However, these new coding tools greatly increase the computational complexity and memory requirement, thus make this standard harder to be adopted. Among them, Motion Estimation (ME) is one of the most powerful tools in terms of coding efficiency, but it also consumes most of the total computation and memory requirement, including memory capacity and memory access [3].

Many works are proposed to achieve the same coding efficiency while keeping the cost down. ME algorithms have been an important research issue for a long time, and using scratch memory as search range (SR) buffer to save external memory bandwidth is also a widely adopted methodology [4]. However, as the video resolution increases, the previous methods described in [4] can hardly keep pace, and solely the SR memory can dominate the chip area [5][6].

In this paper, we introduce a novel cache system and fast ME algorithm. The main considerations of a video coding system, such as rate-distortion performance, computational complexity, on-chip memory size, and external memory bandwidth, are taken care of. Based on the proposed system, the on-chip memory size is greatly reduced, while the other properties are kept in the same level.

The rest of the paper is organized as follows. In section 2, the problem of SR memory is outlined. Section 3 describes the detail of the proposed method and hardware architecture. Section 4 and 5 address our simulation results and conclusion remarks.

2. PROBLEM STATEMENT

Assuming we want to encode a video sequence with 1080p resolution, 60 frames per second, and we use ±128×±64 search range, level C data reuse scheme [4], bi-directional frames (B-frames), then the on-chip SR memory would contain (128×2+16)×(64×2+16)×2 pixels, which occupies 78KB of on-chip SRAM, and the external memory bandwidth would be 2.13GB/s. Nevertheless, for a high-end SoC system running at 200MHz with a fairly wide 128-bit memory bus, the throughput can only achieve 3.2GB/s at 100% bus utilization. As we can see, the bus bandwidth budget is tight even on a high-end SoC system, not to mention the overhead of large on-chip SRAM area.

Previous work shows that the SR utilization is only 30% on CIF video, and it decreases to 15% on D1 video [7]. That is to say, many data read to the SR buffer are never used. Further investigation reveals the trend of low utilization still applies to HD video. However, if we try to save on-chip memory by directly shrinking the search range, the rate-distortion (RD) performance would be greatly hurt. Therefore, a smarter strategy to reduce on-chip memory usage without sacrificing the coding efficiency is desirable.

Moreover, when the video resolution gets higher, the ratio of (cache size/level C buffer size) can be smaller. From the work in [7], if we want a reasonable RD performance, cache size is 1/3 of level C buffer size for D1 video, and the ratio is 2/3 for CIF video. From this trend, we can expect the ratio be smaller on HD video. As a result, it makes more sense to utilize cache-based architecture on HD video coding systems.

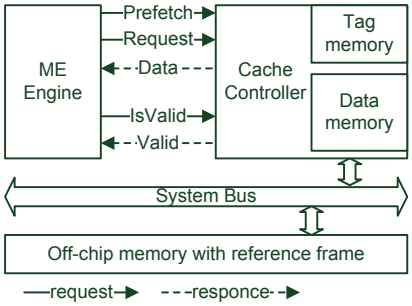

Fig. 1 Hardware architecture of proposed system

3. PROPOSED ALGORITHM AND ARCHITECTURE

In this paper, we introduce a novel cache system, which is dedicatedly designed for the access pattern on reference frames, and a cache-aware fast ME algorithm, which adjusts its own behavior according the state of our cache system. The system architecture is shown in Fig. 1.

3.1 Cache System Design

Since the data access on reference frames are all rectangular in ME stage, so should the addressable unit on cache. In most high-end ME architectures, calculating 8 to 64 candidates per cycle is essential for the Integer Motion Estimation (IME) stage [5], so the memory banks should be abundant and interleaved so adjacent blocks belong to different banks.

In order to gain high bus utilization, transmitting data in burst mode with appropriate length is important. In a typical SoC system, a reading request in burst mode generally takes 20 to 40 cycles. Assuming the bus is 64-bit wide, its peak throughput is 8 bytes per cycle, so it can load a whole 16-by-16 MB in 32 cycles. Thus, this consideration make MB-sized block a suitable shape for a cache line.

In this cache system, the reference frame is addressed by the relative x and y coordinates. The lower bits are used as the address of the cache, and the bits higher than the address space of cache will be used as tags. The set-associativity of this cache system can also vary from directly-mapped to fully associative. We denote the configuration in width × height × ways representation. For example, for a 128×128×2 cache and a 512×512 motion vector (MV) limitation, a MV can be split as follows. The least significant two bits in x and y coordinates are for quarter pixel, the higher 4 bits for the address within a cache line, the higher 3 bits for the address of cache, and the highest 2 bits are the tag. So the cache has 6-bit address and 4-bit tag.

Due to the moving window style of access pattern on the reference frames, we use a static replacement policy that gives left-hand side blocks lower priority because the window moves to the right. If two blocks share the same x-coordinate, then we compare y-coordinate and evict the top one. This static replacement policy makes the architecture simpler because we only need to decide which one to evict when we do a refill, not when we read the data. To accommodate the static replacement policy, the cache should be invalidated when the current MB changes row.

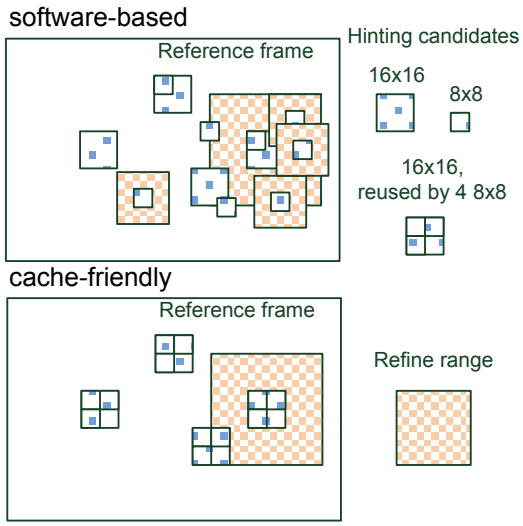

Fig. 2 Memory access pattern of ME algorithms on search range

The hardware of our cache system consists of an on-chip SRAM for actual data, a smaller memory for tags, and a register file for valid bits. The priority is solely dependent on the tags, so no additional information is required.

3.2 Fast ME Algorithm for Cache System

In a software implementation of ME, it is easy to adopt algorithms with lots of data dependencies without huge overhead. This property is derived from the sequential nature of processors and sophisticated general-purposed cache system.

A typical software-based ME algorithm for a single MB is:

1. For all the block types, do step from 2 to 5.
2. Take zero-vector, motion vector predictor (MVP) defined in H.264/AVC, and the final MVs from spatial and temporal neighboring MBs that have already done ME as hinting candidates. If the current block type is sub 16-by-16, then MVs from other sub-blocks within the current MB can also be used.
3. Evaluate the cost of all the hinting candidates; choose the candidate with lowest cost as the origin of refinement.
4. Do a certain type of fast search beginning from the origin of refinement. Find the location with minimum cost within search range.
5. Refine the final result by doing the fractional motion estimation (FME) around the minimum point.

In the given algorithm, the hinting candidates of a sub-block depend on the IME refinement results of other sub-blocks. This data dependency prevents further data reuse and makes prefetching much harder. As a result, we propose a cache-friendly ME algorithm.

As for the data flow, we break the dependency between sub-blocks within the same MB, so the access pattern is more regular, as we can see in Fig. 2. The origin of refinement merely depends on the result of the 16-by-16 MB, and all the sub-blocks just reuse the partial sum of absolute difference (SAD) values. The access pattern would almost be identical to that of the software-based ME without enabling variable block sizes

(VBS), but the RD performance would not be harmed as much, which can be verified in the simulation results.

Cache-awareness is achieved by out-of-order execution of evaluating the hinting candidates. Since the order to evaluate the hinting candidates doesn't matter, the ME core could choose the ready candidates first, while prefetching other candidates in the background.

3.3 Prefetching Algorithm

Designing and evaluating prefetching algorithm require accurate timing information, so the MB-pipeline we used will be introduced briefly first. Four stage MB pipeline is adopted in our work, and they are: IME, FME, Intra coding/Motion Compensation (MC), and deblocking/entropy coding [8]. According to our cache-aware fast ME algorithm, it is easier to do data prefetching. Since the hinting points are from the neighboring MBs, the latest hint would be available when the left-hand side MB finishes its IME refinement, which is around the same time as current MB starts its IME stage. Accordingly, the hinting point from the left MB is the only hint that might not be prefetched soon enough, but the PEs can process other hinting points first as described in previous subsection. On the other hand, the origin of refinement does depend on the cost of all the hinting points, and they are all in the same IME stage. To resolve this issue, we speculate the result, and do prefetching over the speculated refinement range.

4. SIMULATION RESULTS

4.1 Experimental setup

In this paper, we use a software platform to encode H.264/AVC video. The testing raw sequences are all in 720p and 1080p resolution.

The parameters of the encoder are listed below.

Num. of reference frames	1
Group of Picture (GOP) structure	I(bBbP)*, i.e. P-frame period is 4
Total frames	33 (8 GOPs)
Quantization parameter (Qp)	20, 21, 30 and 31
Entropy coder	CABAC
High complexity mode	Disabled
Sub 8x8 modes	Disabled in proposed algorithm
ME algorithm	Full search
Refinement range	16 pixels

Table 1. Parameters of encoder

Only the first frame is encoded as intra-frame, and only P-frames are used to analyze the bandwidth. The two consecutive Qps are used for RD curve interpolation, so that the PSNR drop can be fairly compared at the same bitrate. Sub 8x8 modes are disabled because it is not very helpful in HD video. Figures of 720p or Qp 20 are not listed here due to the limited space. Results with Qp 20 and Qp 30 give the same conclusion.

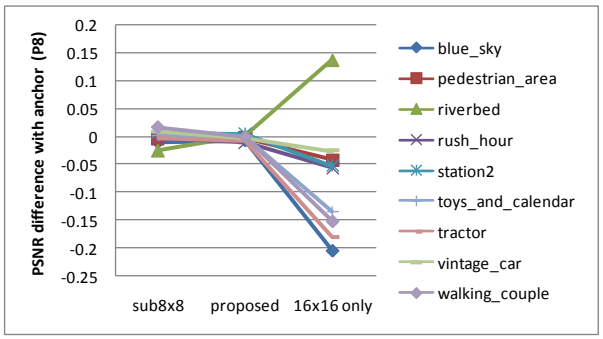

Fig. 3 Rate-distortion performance of different ME algorithms

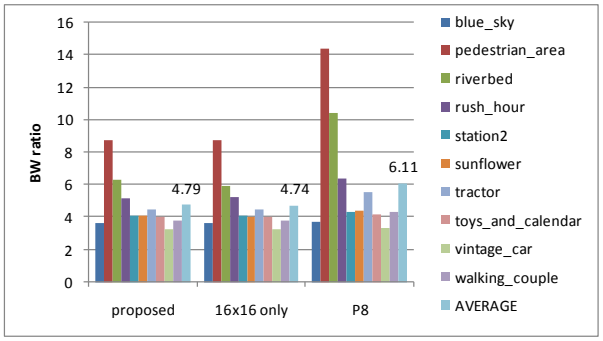

Fig. 4 External bandwidth of different ME algorithms on 64x64x2 cache

4.2 Explorations

Four ME algorithms, software-based with all VBS modes (labeled as sub8x8), software-based without sub 8-by-8 blocks (P8), software-based without any VBS modes (16x16 only), and cache-based algorithm (proposed), are compared.

The size of SR memory of level C data reuse scheme for one reference frame is 38KB, so cache size of 32KB is used as an upper bound reference. ME refinement with 16-pixel search range requires a 48-by-48 pixel buffer, and that is already larger than 2KB, so 4KB is our lower bound of cache size. We denote the configuration in width × height × ways representation. The chosen four configurations are 64x64, 128x64, 64x64x2, and 256x128, and the sizes are 4KB, 8KB, 8KB, and 32KB respectively.

Bandwidth ratio is defined as (bandwidth requirement)/(current frame size). For example, bandwidth ratio 9 means the bandwidth spent on loading the reference frame is 9 times larger than loading the current frame.

4.3 Results

We show the PSNR drop of four different ME algorithms in Fig. 3. The Y-axis is the PSNR difference with the anchor, P8, so P8 is not plotted. It is obvious that enabling sub8x8 modes is not useful, and using the cache-based algorithm doesn't affect the PSNR much.

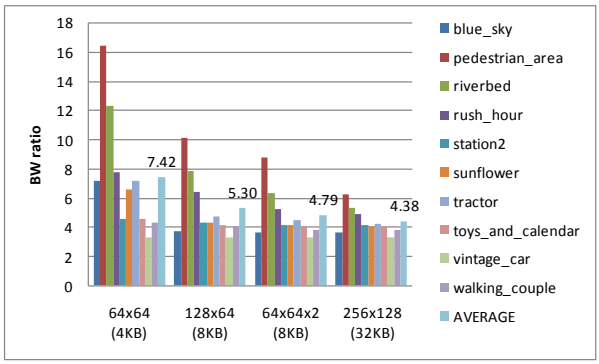

Fig. 5 External bandwidth of proposed ME algorithm on different cache architectures

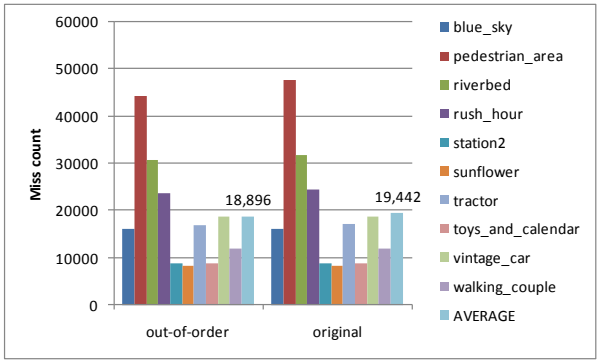

Fig. 6 Miss count reduction in the hinting point evaluation stage by out-of-order execution

The external memory bandwidth of three ME algorithms are provided in Fig. 4. The cache configuration used here is 64×64×2. The bandwidth of P8 is 28% higher than proposed on average. Combining Fig. 3 and Fig. 4, we can see the proposed cache-based ME algorithm uses bandwidth as small as "16x16 only", while keep similar quality as "P8".

The external memory bandwidth of four different cache architectures are shown in Fig. 5. With vertical ±64 searching range, the bandwidth ratio of level C data reuse scheme is 9. Since the refinement range is 16 pixels, the bandwidth ratio has a lower bound of 3 when the whole frame shares the same MV. From this figure, bandwidth ratio of configuration 64×64×2 is always smaller than 9, and the average is 4.79, proving the effectiveness of our cache architecture. For 720p sequences, the average is 3.93, and the maximum is 6.37. In addition, out-of-order execution can further decrease the misses in the hinting point evaluation stage. The result shown in Fig. 6 is the comparison of proposed algorithm with and without out-of-order execution, and the cache configuration is 64×64×2. The average miss count is reduced by 3% after enabling out-of-order execution.

5. CONCLUSION

In this paper, the algorithm and architecture of cache system and fast ME for HD H.264/AVC are proposed. Compared with software-based algorithm, the proposed cache-friendly ME algorithm running on top of the proposed cache system reduces the external memory bandwidth by 8.5% and 21% for 720p and 1080p sequences on average, while the rate-distortion drop is less than 0.03dB and 0.02dB. Comparing the cache-based system and the original level C data reuse scheme, the on-chip memory size is reduced to 21%, and the external memory bandwidth is also decreased by 56% and 47% for 720p and 1080p sequences on average. For a cache with only 10% the level C size, the bandwidth reduction is 50% and 18% respectively. Out-of-order execution further decreases the miss count of hinting candidates by 3%.

6. REFERENCES

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, *ITU-T Rec.H.264 and ISO/IEC 14496-10 AVC*, Joint Video Team, May 2003.

[2] Wiegand, T., Sullivan, G.J., Bjntegaard, G., Luthra, A., "Overview of the H.264/AVC video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.13, no.7, pp. 560-576, July 2003

[3] YH. Chen, T.C. Chen, and L.G. Chen, "Hardware oriented content-adaptive fast algorithm for variable blocksize integer motion estimation in H.264," *Proc. Int. Symp. on Intell. Signal Processing and Commun. Syst. (ISPACS)*, pp. 341-4, 2005.

[4] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 61–72, Jan. 2002.

[5] Z. Liu, Y. Song, M. Shao, S. Li, L.F. Li, S. Ishiwata, M. Nakagawa, S. Goto, T. Ikenaga, "A 1.41W H.264/AVC REAL-TIME ENCODER SOC FOR HDTV1080P" *2007 Symposium on VLSI Circuits* Digest of Technical Papers

[6] Y. W. Huang, et al., "A 1.3TOPS H.264/AVC single chip encoder for HDTV applications," *ISSCC Dig. Tech. Papers*, pp. 128-129, Feb. 2005.

[7] C.Y. Tsai, C.H. Chung, Y.H. Chen, T.C. Chen, L.G. Chen, "Low Power Cache Algorithm and Architecture Design for Fast Motion Estimation in H.264/AVC Encoder System," *ICASSP 2007.* IEEE International Conference, vol.2, no., pp.II-97-II-100, 15-20 April 2007

[8] T.C. Chen, Y.W. Huang, L.G. Chen, "Analysis and design of macroblock pipelining for H.264/AVC VLSI architecture," *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on* , vol.2, no., pp. II-273-6 Vol.2, 23-26 May 2004