HD VIDEO DECODING SCHEME BASED ON MOBILE HETEROGENEOUS SYSTEM ARCHITECTURE

Yu-Jung Chen, Yu-Sheng Lin, Hsin-Fang Wu, Chia-Ming Chang, Shao-Yi Chien

Graduate Institute of Electronics Engineering, National Taiwan University
1, Sec. 4, Roosevelt Rd., Taipei 106, Taiwan, R.O.C.

ABSTRACT

Efficiency for mobile devices becomes important than ever due to the demanding multimedia applications. Take HD video decoding for example, it is getting challenging as the resolution increases. To enhance efficiency, HSA (*Heterogeneous System Architecture*) is proposed to drive a new era of computation. Based on this idea, we develop a heterogeneous architecture, which is composed of a mobile GPU (*Graphics Processing Unit*) with a proposed configurable filtering unit and a CPU to coordinate the decoding flow. The feasibility of H.264/MPEG-4 HD video decoding pipeline for the proposed architecture is verified. Furthermore, according to the corresponding GPU hardware model, the estimated processing time of inverse quantization, inverse transform and motion compensation for decoding a 1080p video can reach to 16ms and 11ms per frame for MPEG-4 and H.264 respectively.

***Index Terms*—** Heterogeneous computing, GPU, video decoding

1. INTRODUCTION

Nowadays, owing to the advances in programmable graphics hardware, GPUs (*Graphics Processing Units*) have become indispensable either for dedicated rendering applications or general purpose computing workload. To improve the communication and control flow between CPU and GPU, several SoC IP vendors, including AMD, ARM, Qualcomm, MediaTek, Samsung and etc., organize HSA (*Heterogeneous System Architecture*) foundation to establish heterogeneous computing architecture. The idea behind HSA is quite similar to AMD Fusion on desktop system, as illustrated in Fig. 1. In this architecture, CPU and GPU are tightly-coupled, and communicate through shared memory space. Regarding to architectural features, CPU excels in serial control flow and multi-task parallelism; GPU outperforms in vector processing to gain data parallelism. Aimed at demanding workloads, such heterogeneous computing architecture can exploits both features to deliver processing efficiency.

As the storage and network bandwidth grow, HD video decoding turned out to be a fundamental entertaining application for mobile devices. However, decoding HD video contents in real-time is challenging for mobile devices. In this work, we simulate a heterogeneous computing architecture for mobile devices. To exploit the multiprocessor parallelism in GPU, CPU is responsible for coordinating decoding flow and dispatching threads. Decoding stages, such as inverse quantization and inverse transform, are scheduled as parallel vector threads for GPU. Moreover, a proposed modified configurable filtering unit is used for supporting fractional motion compensation, which is proven to be time-consuming in a decoding pipeline. Considering the efficiency, the decoding stages, which are not suitable for GPU (e.g., entropy decoding and deblocking filtering), are left for CPU to simulate. To quantify performance, the corresponding GPU RTL model is synthesized with TSMC 65nm technology and operating frequency can reach to 300MHz. The estimated processing time of inverse quantization, inverse transform and motion compensation in 1080p video decoding process can reach to 16ms and 11ms per frame for MPEG-4 and H.264 respectively.

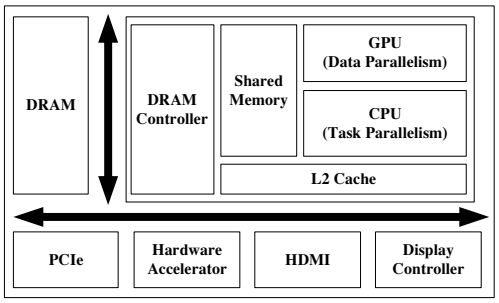

Fig. 1. Simple illustration of AMD Fusion.

2. RELATION TO PRIOR WORKS

2.1. Prior Works

As the coding standard evolves, algorithm complexity grows to achieve better visual quality. Real-time requirement is harder to meet. Besides pure ASIC solutions, researchers accelerate video coding in various processor architectures. *General-Purpose Computation on GPU* (GPGPU) is a popular means of accessing the processing power in commodity

GPUs through programming models such as shading languages (e.g., Cg, OpenGL and DirectX), OpenCL and CUDA. A well-studied problem in video codec for GPGPU is motion estimation (ME), which is the most demanding stage of the codec. Thread scheduling and memory bandwidth utilization for ME are highly optimized with CUDA[1][2]. In addition, MPEG-2 decoding pipeline has been accelerated and adapted to conventional graphics pipeline with OpenGL and Cg [3].

From the perspective of processor architecture, Chien et.al [4] propose a video accelerating instruction set and configurable memory array to reduce off-chip memory bandwidth for multimedia stream processor. Targeting at variable sized memory block fetch and ALU utilization, Lo et.al [5] propose an improved SIMD (*Single Instruction Multiple Data*)-based video processor. Besides, *Cell Boardband Engine* is a specific heterogeneous processor architecture which is composed of SIMD-based architecture—SPE (Synergistic Processing Element), and PPE (Power Processor Element). Numerous implementations and techniques for accelerating video decoding pipeline are discussed and developed [6][7][8].

2.2. Comparison with Prior Works

We summarize the features of prior works and classify them into three categories.

(a) Optimize threads scheduling and memory bandwidth utilization on GPUs [1][2][3].

(b) Microarchitecture design and implementation [4][5].

(c) Implementation and performance optimization on heterogeneous processor architecture [6][7][8].

Compared with prior works, our proposed architecture is actually featured with these three categories. First, the vector thread scheduling of GPU is tailored for the decoding pipeline. Second, the microarchitecture of configurable filtering unit is modified from conventional *texture fetch* instruction. Third, the proposed decoding pipeline is based on general heterogeneous system architecture, which is composed of CPU and GPU. Such general heterogeneous architecture would probably dominate future computing trend.

3. PROPOSED VIDEO DECODING SCHEME

3.1. MPEG-4/H.264 Decoding Pipeline

As shown in Fig. 2, a typical decoding pipeline is composed of entropy decoding, inverse quantization, inverse transform, motion compensation and deblocking filtering. For MPEG-4, after acquiring decompressed video stream through entropy decoding (VLD), the decoder executes inverse quantization which defines two modes—H.263 and MPEG-4 mode. Eq.(1) shows the H.263 mode, where a 5-bit integer iQ is associated with variable bitrate, and $in[i]$ represents the i-th coefficient

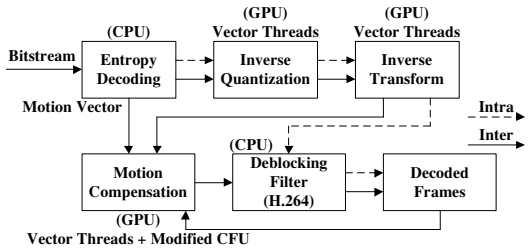

Fig. 2. A typical video decoding pipeline (with H.264 in-loop deblocking filter).

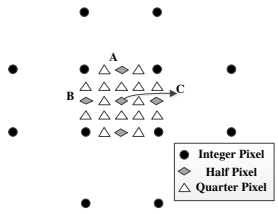

Fig. 3. Sub-pixel motion compensation

of the decompressed macroblock after VLD. For MPEG-4 mode, an extra quantization table mat is to provide better quality. As eq.(2) shows, (a) is for inter macroblock and (b) is for intra macroblock. Next, a general 8×8 2-D IDCT is applied as inverse transform. Besides integer-pixel motion compensation, sub-pixel motion compensation is supported to half-pixel precision. Last, a post-processing deblocking filter is optional for enhancing visual quality.

$$2 \times iQ \times in[i] + 2\lfloor iQ/2 \rfloor \tag{1}$$

$$\begin{cases} (in[i] \times mat[i] \times iQ)/8 & \text{(a)} \\ [(2in[i] + \text{sgn}(in[i])) \times mat[i] \times iQ]/16 & \text{(b)} \end{cases} \tag{2}$$

H.264 defines advanced features than MPEG-4. Compared with MPEG-4, the adopted entropy coding technique in H.264, such as Context Adaptive Variable Length Decoding (CAVLD), has higher coding performance. For levels of inverse quantization, H.264 has six pre-defined tables, and these tables are similar to mat defined in MPEG-4. Besides 2-D DCT, H.264 adopts integer transform, which requires only addition, subtraction and bit-shifting, has lower computation effort.

Sub-pixel motion compensation is also adopted in both MPEG-4 and H.264 to enhance visual quality, as shown in Fig.3. To get half-pixels, MPEG-4 uses bilinear filter, while H.264 utilizes a six-tap filter for either horizontal or vertical direction (**A** and **B** in Fig.3). The centering half-pixel (**C** in Fig.3) are then obtained by applying the same filter in both directions. Furthermore, quarter-pixels are acquired by the rounded average of two neighboring pixels. Finally, H.264 adopts an in-loop deblocking filter, and the filtered results are later used for motion compensation.

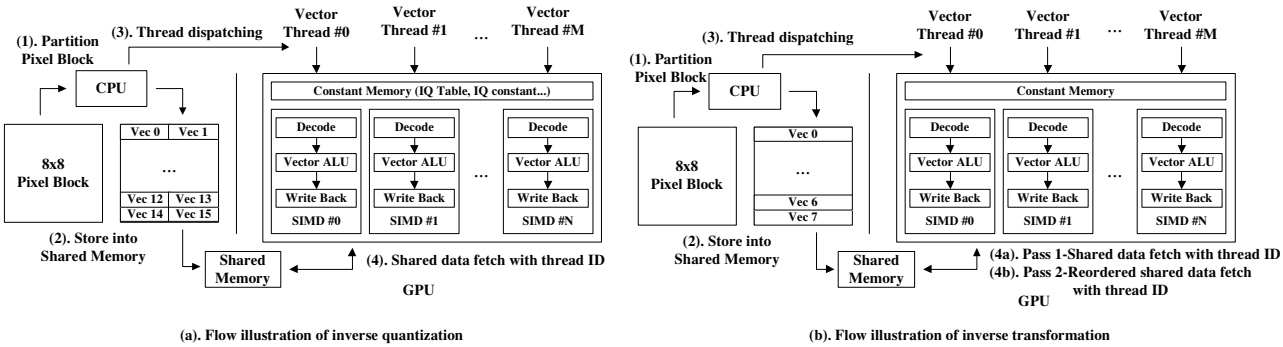

Fig. 4. Illustration of MPEG-4 inverse quantization (IQ) and inverse transform (IDCT).

3.2. Proposed Techniques

The decoding pipeline is partitioned into stages for either GPU or CPU as Fig.2 shows. According to characteristics of computation for each stage, we discover that inverse quantization, inverse transformation and motion compensation are efficient in SIMD architecture. Based on our developed GPU architecture, which is composed of four-lane SIMD processors, CPU can dispatch parallelized vector threads to gain both *thread parallelism* (multiprocessors) and *data parallelism* (SIMD microarchitecture). These SIMD processor are equipped with our previous proposed *Configurable Filtering Units* [9] with dedicated modifications for motion compensation.

3.2.1. Inverse quantization–IQ

We categorize inverse quantization into two types of operations. **Type I**: Manipulating decompressed pixel blocks with a constant value; **Type II**: Manipulating decompressed pixel blocks with an inverse quantization table. Take MPEG-4 as an example, Fig. 4 (a) illustrates how we utilize the vector operation. The 8×8 pixel block is first divided to sixteen *vec4* (four-component vectors), and each vector is dispatched to a SIMD processor. Based on the stream type, CPU is responsible for preparing corresponding coefficients for IQ, either a pre-defined table or merely constant values. The number of dispatched vector thread can be configured, and scheduled by CPU. That is, if we have sixteen SIMD processors, 8×8 pixel block for MPEG-4 inverse quantization can be completed at once. The mechanism for H.264 inverse quantization is similar except the block size is changed to 4×4 and the block is divided to four *vec4*.

3.2.2. Inverse transform–IT

For MPEG-4, a two-pass butterfly structured [10] IDCT is adopted in this work. If we viewed the block as an 8×8 matrix, the IDCT can be described as follow: first, the 1D-IDCT is applied to each row, and then the 8×8 block are transposed. The inverse transformed result is obtained after applying the procedure again. Fig. 4 (b) illustrates the proposed data flow with a four-lane SIMD architecture. First the 8×8 block is divided into eight rows and each row is processed concurrently, where eight coefficients are divided into two four-coefficient bundles. With a four-channel SIMD processor, these scalar operations can be executed at once.

Instead of the 8×8 IDCT block size in MPEG-4, H.264 adopts inverse integer transform with smaller block size (4×4). This yields simpler coefficients and is easier to calculate. Shift operation can replace the multiplication. In the inverse integer transform, the de-quantized macroblock needs to be divided into several 4×4 sub-blocks first for Hadamard matrix. The scheduling scheme is similar to MPEG-4. The 4×4 sub-block is divided to four four-component vectors. Each row/column operation requires only one component-wise shift and summation to acquire output.

3.2.3. Motion compensation–MC

In our developed architecture, texture filtering is performed with a texture unit to access texels from the texture buffer by a configurable filtering unit [9] with various general filtering patterns. For sub-pixel motion compensation, we proposed a modified configurable filtering unit to enhance the performance. However, the filtering coefficients differ according to the sub-pixel coordinates, especially for quarter-pixels. As previously mentioned, each quarter-pixel is obtained through averaging two neighboring pixels. These two neighboring pixels can either be integer- or filtered half-pixel depending on the quarter-pixel coordinates. Due to various coefficient configurations for sub-pixel coordinates, directly obtaining quarter-pixel among parallel threads is inefficient. Consequently, the modified configurable filtering unit can fetch either integer- or half-pixel coordinate with specified filtering coefficients. Fig. 5 illustrates our configuration of the proposed filtering unit, a 6-tap symmetric filter is employed to the neighboring pixels. Applying corresponding filtering coefficients, filtered sub-pixel either in H.264 or MPEG-4 can be obtained through this architecture.

While H.264 defines quarter-pixel motion compensation,

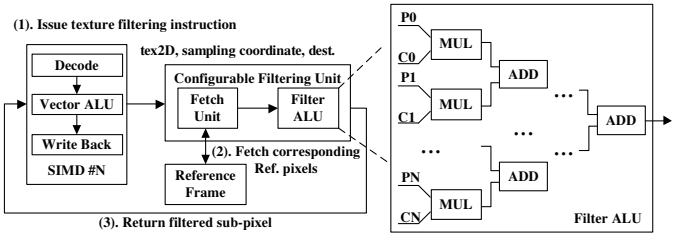

Fig. 5. Configuration of proposed modified configurable filtering unit.

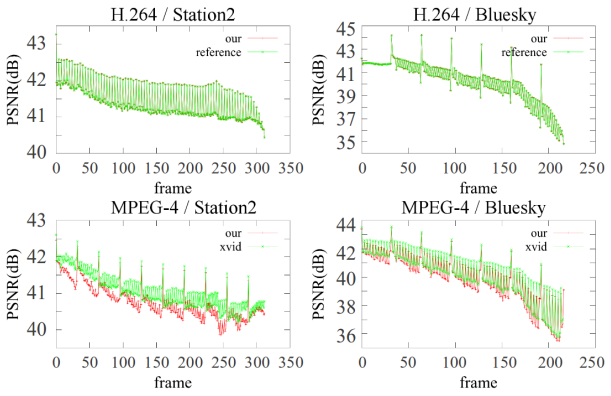

Fig. 6. The simulated PSNR result for H.264 BP and MPEG4 SP.

the fetching process is turned into a two-pass filtering procedure in the proposed architecture. In the first pass, one of the required half-pixel is first fetched, and the filtered value is stored in registers. In the second pass, another filtered half-pixel is fetched. After averaging and rounding, filtered quarter-pixel value is obtained.

4. EVALUATION

Based on the proposed architecture, we simulate and compare visual quality of our implementation with Xvid and H.264 JM reference software. To evaluate the decoding quality with these standards, we encode the video sequence by using reference software first and decode the compressed bitstream through the reference software and our software-simulated architecture respectively. Fig. 6 shows the PSNR results for 1080p HD video sequence, and the intra-frame refresh rate is 32. MPEG-4 simple profile (SP) and H.264 baseline profile (BP) is adopted for evaluation. It shows that our implementation of H.264 suffers much less PSNR drop than that of MPEG-4. The main reason for the PSNR drop is encoder/decoder mismatch in inverse transform. Modern codecs have adopted fixed-point transform for speed consideration. Xvid MPEG-4 codec adopts an asymmetric fixed-point IDCT approximation. However, we adopt symmetric pure floating-point operation to fit our SIMD processor. The rounding error is raised from the floating-point arithmetic units and causes inverse transform mismatch.

To evaluate the performance on our proposed architecture, the total execution time of inverse quantization, inverse transform and motion compensation are estimated by the hardware model. Because entropy decoder and deblocking filter are simulated with CPU, the execution time for these two stages is not included. The estimated execution time can illustrate the margin left for achieving real-time requirement. Due to the configurable task dispatching, the number of SIMD processor can be scaled to boost performance. Our SIMD processor architecture has a corresponding RTL model and synthesized with TSMC 65nm technology. The operating frequency can reach to 300MHz. For decoding videos in 1080p, the estimated execution time for MPEG-4 and H.264 are shown in Table. 1. The maximum speed-up factor depends on the number of threads we use. It also shows that the speed-up factor saturates when we scale the number of SIMD processors to 16.

Codec	Number of cores		
	4	8	16
H.264	68.1%	34.1%	23.4%
MPEG-4	95.4%	47.7%	37.5%

Table 1. Estimated 30fps operation percentage of proposed architecture versus the number of SIMD processors for decoding a 1080p frame. Because the upper bound of decoding a video block is 8, the decoding time can be reduced by 50% while the number of SIMD cores is increased from 4 to 8. However, if we further increased the number of cores to 16, the efficiency boost will only be 22 and 31% for H.264 and MPEG4, respectively.

5. CONCLUSIONS

Previous similar works mainly focused on desktop GPU. In this work, we shows that a mobile GPU can support 1080p video without a dedicated video decoder. By simulating a heterogeneous computing composed of a mobile GPU with proposed configurable filtering unit and a CPU for coordinating decoding flow, we verify the compatibility and visual quality of the proposed architecture. The estimated processing time of inverse quanti- zation, inverse transform and motion compensation for 1080p video decoding can reach to 16ms and 11ms per frame for MPEG-4 and H.264 respectively with eight SIMD processors configuration.

6. REFERENCES

[1] Shang-Te Yang, Tsung-Kai Lin, and Shao-Yi Chien, "Real-time motion estimation for 1080p videos on graphics processing units with shared memory optimization," in *Signal Processing Systems, 2009. SiPS 2009. IEEE Workshop on*, oct. 2009, pp. 297 –302.

[2] Nagai-Man Cheung, Xiaopeng Fan, O.C. Au, and Man-Cheung Kung, "Video coding on multicore graphics processors," *Signal Processing Magazine, IEEE*, vol. 27, no. 2, pp. 79 –89, march 2010.

[3] Bo Han and Bingfeng Zhou, "Efficient video decoding on gpus by point based rendering," in *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, New York, NY, USA, 2006, GH '06, pp. 79–86, ACM.

[4] Shao-Yi Chien, You-Ming Tsao, Chin-Hsiang Chang, and Yu-Cheng Lin, "An 8.6 mw 25 mvertices/s 400-mflops 800-mops 8.91 mm multimedia stream processor core for mobile applications," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 9, pp. 2025 –2035, sept. 2008.

[5] Wing-Yee Lo, D.P. Lun, Wan-Chi Siu, Wendong Wang, and Jiqiang Song, "Improved simd architecture for high performance video processors," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 12, pp. 1769 –1783, dec. 2011.

[6] Chi Ching Chi, Ben Juurlink, and Cor Meenderinck, "Evaluation of parallel h.264 decoding strategies for the cell broadband engine," in *Proceedings of the 24th ACM International Conference on Supercomputing*, New York, NY, USA, 2010, ICS '10, pp. 105–114, ACM.

[7] Michael A. Baker, Pravin Dalale, Karam S. Chatha, and Sarma B.K. Vrudhula, "A scalable parallel h.264 decoder on the cell broadband engine architecture," in *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, New York, NY, USA, 2009, CODES+ISSS '09, pp. 353–362, ACM.

[8] Yongjin Cho, Seungkyun Kim, Jaejin Lee, and Heonshik Shin, "Parallelizing the h.264 decoder on the cell be architecture," in *Proceedings of the tenth ACM international conference on Embedded software*, New York, NY, USA, 2010, EMSOFT '10, pp. 49–58, ACM.

[9] Chih-Hao Sun, Ka-Hang Lok, You-Ming Tsao, Chia-Ming Chang, and Shao-Yi Chien, "Cfu: multi-purpose configurable filtering unit for mobile multimedia applications on graphics hardware," in *Proceedings of the Conference on High Performance Graphics 2009*, New York, NY, USA, 2009, HPG '09, pp. 29–36, ACM.

[10] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck, *Discrete-Time Signal Processing*, Prentice-Hall, 1984.