A 16-65 CYCLES/MB H.264/AVC MOTION COMPENSATION ARCHITECTURE FOR QUAD-HD APPLICATIONS

Jinjia Zhou, Dajiang Zhou, Gang He, and Satoshi Goto

Graduate School of Information, Production and Systems, Waseda University.
2-7 Hibikino, Kitakyushu 808-0135, Japan.
E-mail: zhou@ruri.waseda.jp

ABSTRACT

This paper presents a motion compensation architecture for Quad-HD H.264/AVC video decoder. For meeting the high throughput requirement, reducing power consumption and solving the memory latency problems, three optimization schemes are applied in this work. Firstly, a quarter-pel interpolator based on Horizontal-Vertical Expansion and Luma-Chroma Parallelism (HVE-LCP) is proposed to efficiently increase the throughput by at least 4 times from the previous designs. Secondly, a novel cache memory organization (4Sx4) is adopted to improve the on-chip memory utilization, contributing to memory area and power saving. Finally, a Split Task Queue (STQ) architecture enhances the memory system latency tolerance, which reduces overall processing time. This design costs a logic gate count and on-chip memory of 108.8k and 3.1kB, respectively. The proposed architecture supports real-time processing of 3840x2160@60fps at 166MHz.

1. INTRODUCTION

While 1080 HD has already become a current standard for TV broadcasting and home entertainment, even higher specifications such as 4Kx2K Quad-HD format, have been targeted by next-generation applications. To store and transmit these mass video contents, video compression is indispensable. Compared with previous MPEG standards, H.264/AVC provides over two times higher compression ratio with better video coding quality, which makes it a promising tool for compression these massive data. The high coding efficiency of H.264/AVC comes from various new features, such as variable block size motion compensation, quarter-sample fractional interpolation, multi-mode intra prediction, context adaptive entropy coding and so on. However, the use of these new techniques, along with the ever-increasing demand for resolution, greatly challenges the design of video decoders. The 4kx2k motion compensation(MC), which is speed bottleneck of the whole decoder, is mainly challenged by the following aspects.

Firstly, compared with HD application, the throughput requirement for MC interpolation in Quad-HD cases is increased by at least 4 times. To meet this requirement, the straightforward way is to process four rows in parallel instead of one row as previously proposed in [8]. Although the parallelized architecture can increase the throughput, the critical data alignment problem will lead to extra overhead of both the memory read power and interpolation processing time. This means the cost of parallelism will be larger than the enhancement in throughput.

Secondly, with higher specifications, memory bandwidth requirement increases significantly. [2] optimized the bandwidth for motion-compensated temporal filtering, which is utilized for scalable video coding. [1][3][5] proved that cache system can be an effective way to reduce the external DRAM bandwidth for the general motion compensation. However, the on-chip memory bandwidth from the cache system to the interpolation component becomes higher and costs larger power consumption, because the width of data memory increases proportionally with the interpolation parallelism.

Thirdly, the latency between cache sending the request to receiving the data from the memory system becomes longer due to two reasons. One is that the DRAM latency increases because of higher-speed DRAM specifications such as DDR2 and DDR3. On the other hand, new techniques adopted to enhance the DRAM access efficiency, such as reference frame recompression [10], though reduces the total access amount, incurs longer access delay. As a result, while the memory system latency is only around 10 clock cycles in HD decoders, it can increase to over 40 clock cycles in the new Quad-HD applications. Generally, to hide the DRAM latency, task queue is utilized in a cache system. However, this architecture requires conflict checking to avoid flushing the useful data in the cache, which will be described in Section 3. The longer memory system latency will drastically increase the probability of conflict in the cache system, which results in long pipeline stall and decreases the overall system performance.

To solve the above issues and achieve an efficient MC architecture for H.264/AVC real-time decoding of Quad-HD applications, three schemes are proposed in this paper. Firstly, Horizontal-Vertical Expansion and Luma-Chroma Parallelism (HVE-LCP) based interpolation is implemented to reduce the influence from data alignment problem while increasing the decoding throughput to at least over 4 times as the previous works. Secondly, an efficient cache memory organization scheme (4Sx4) is adopted to improve the on-chip memory utilization. By applying this scheme, memory area is reduced and memory power is saved by 39%∼49%.

Finally, by employing a Split Task Queue (STQ) architecture, the cache system becomes capable of tolerating much longer latency of the memory system. Consequently, the cache idle time is saved by 90%, which contributes to reducing the overall processing time by 24%∼40%. The remainder of this paper is organized as follows. Section 2 and Section 3 describe the proposed design for the interpolation and cache components. Implementation results and conclusion are given in Section 4 and Section 5, respectively.

2. PARALLELISM OF MC INTERPOLATION

Most of the previous works on MC interpolation decompose an MB (Macroblock)into 16 4x4 blocks and for each 4x4 block load an area of at most 9x9 reference pixels. As described in [8], 4 pixels in the same row are processed simultaneously to improve the data reuse and reduce the processing time.

However, the 4x4 block based row by row interpolation requires at most 288 clock cycles for processing one MB, which can not meet the requirement of 4Kx2K application. To increase the throughput, one solution is to expand the row of 4 pixels to 8 pixels (horizontal expansion), as shown in Figure 1 (a). However, when the partition size for inter prediction is 4x4 or 4x8, this method does not seem efficient. Since the 8 pixels in one row are from two different partitions, there are no loading data that can be shared and the processing speed can not be improved. Moreover, when expanding one row from 8 pixels to 16 pixels, this method results in almost no improvement on the throughput. Another way to increase the throughput is to process two or more rows in parallel (vertical expansion), as shown in Figure 1 (b). The processing time of 4x4 and 4x8 sized partitions, can also be shortened when using the vertical expansion method. However, the data alignment problem will decrease the speed. Especially for 4Kx2K applications, when four lines are par-

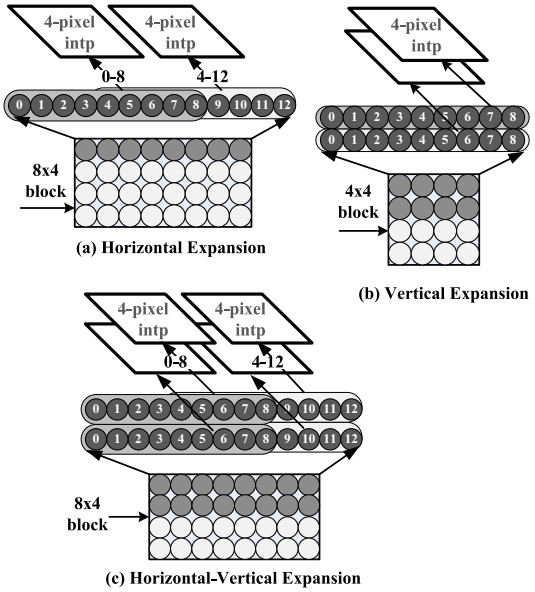

Figure 1: Interpolation parallelism analysis.

allelized, the data alignment problem becomes more serious. For example, loading a vertically unaligned 4x4 block requires 2 clock cycles even when each word stores a 4x4 block. More loading clock cycles will not only increase the memory power but also decrease the processing speed. Another parallelization method for the interpolation is to process two 4x4 blocks simultaneously, which is employed by Sze et al. [7]. However, the corresponding internal memory organization and data control can be very complicated.

To obtain a suitable parallelization method for 4Kx2K application, we propose to combine the horizontal and vertical expansion methods based on the following considerations. Firstly, regarding the high-level limits described in the H.264/AVC standard, although the horizontal expansion method is not so efficient for 4x4 and 4x8 partitions, it will not influence the average speed. H.264/AVC standard defines that the specification higher than or equal to 720x576@25fps, bi-prediction Motion Vector (MV) is not allowed for partition sizes smaller than 8x8. This means the data loading times for interpolation of 8x4, 4x8 and 4x4 partitions can be less than that of the larger ones. The other one is that on levels higher than 3.1, maximum number of motion vectors per two consecutive MBs is 16, which further constrains the influences of small blocks. Moreover, a 4Sx4 internal memory organization, which is to be introduced in Section 3.2, can be utilized for the horizontal expansion method to reduce the memory data width. However, 8-pixel-parallel processing still can not meet the throughput requirement of 4Kx2K application. Therefore, based on the horizontal expansion, a vertical expansion is further applied to process 2 rows in parallel, as shown in Figure 1 (c). Compared with the 4-row-parallel vertical expansion, the memory width of the proposed horizontal-vertical expansion method is reduced by half, and the influence from alignment problem is decreased.

Moreover, in order to further enhance the throughput, the interpolation of luma and chroma samples are parallelized. Since it was originally not easy to reuse the hardware resources of luma and chroma interpolation components, the luma-chroma parallelism can provide 1.5 times the performance (for 4:2:0 sampling) with almost no hardware cost overhead.

Consequently, compared with the general 4x4 block based row by row interpolation architecture, the proposed Horizontal-Vertical Expansion and Luma-Chroma Parallelism (HVE-LCP) based interpolation can enhance the throughput to at least over 4 times.

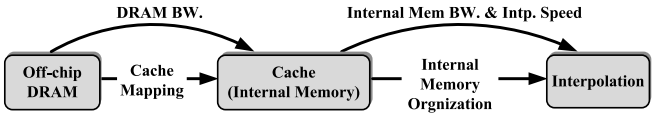

Figure 2: Cache mapping and internal memory organization.

3. PROPOSED CACHE ARCHITECTURE

As shown in Figure 2, for the cache system design, the cache mapping is targeted to reduce the off-chip DRAM bandwidth, and the internal memory organization is aimed to improve the data throughput and save the on-chip memory bandwidth. Cache mapping has been well discussed in previous contributions, but few works pay much attention to the internal memory organization. In a 4Kx2K cache system, in order to meet the higher data throughput requirement, the width of internal memory should increase proportionally. Moreover, the data alignment problem introduced in Section 2 further increases the bandwidth of internal memory bandwidth. In the meanwhile, with a higher parallelism, the area increase of the other parts of the decoder is usually smaller than the speed-up [10]. Therefore, the power and cost portion of the internal memory part becomes more significant in the whole decoder system, if a more efficient memory organization is not proposed. The cache mapping method of this work is given in 3.1, and the proposed internal memory organization is presented in 3.2.

Moreover, the memory system latency is increased from around 10 clock cycles in HD decoders to over 40 clock cycles in the new Quad-HD applications. The longer task queue is required to hide the longer memory system latency, and the longer task queue will drastically increase the probability of conflict in the cache system, which results in long pipeline stall and decreases the overall system performance. Therefore, the general one task queue based conflict checking mechanism is no longer efficient for the longer system memory latency. The detail of the problem and the proposed solution are discussed in 3.3.

3.1 2-D Cache Mapping

Reference read operation of motion compensation (MC) composes a dominant portion of a video decoder's DRAM traffic. To reduce this part of DRAM bandwidth, cache based architecture is utilized for reusing the overlapped reference samples of neighboring blocks. Figure 3 (a) shows the 2-set 2x2-MBs sized 2-D cache for this work, which is similar to the design in [1]. The 2-D organization combines the lower parts of the parX and parY physical coordinates of the Access Units (AUs), which are the basic storage units in the DRAM, to be the cache index. The higher parts of parX and parY coordinates, together with the picture ID (used to specify the physical storage slot of a decoded frame in the DRAM) are combined to be the tag. Considering the use of bi-directional inter prediction in the latest video coding standards, two cache sets should be required for the two reference lists respectively. In our 4Kx2K video decoder [10], because of a wider BUS width and the use of frame recompression technique, AU size equals to the compression unit size, which is larger than that in [1]. The other difference from [1] is that the luma and the corresponding chroma samples are combined into the same AU. Hence, in this work, the AU size is 384 bits containing the luma and chroma samples of an 8x4 block in the reference frame, as shown in Figure 3 (b). Moreover, Partial-MB reordering (PMBR) applied in our whole decoder [10] can increase the cache hit ratio. For the MC cache architecture, PMBR is only related to the cache size. In this paper, to make a fair comparison with the other works, we use a non-PMBR configuration of the cache. As a result, by applying the 2-D cache mapping, an average of 60% reduction of external DRAM bandwidth for reference frame read can be achieved, on the bases of the previous VBSMC [6] scheme.

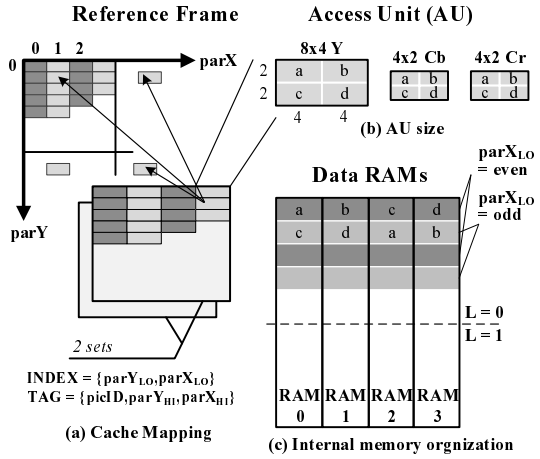

Figure 3: Cache memory design.

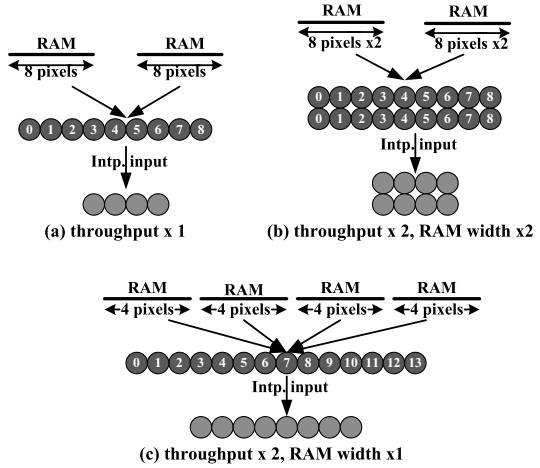

Figure 4: Different internal memory organization analysis.

3.2 Internal Memory Organization

The proposed internal memory organization is targeted to meeting the high data throughput requirement from interpolation, while not significantly increasing memory area and memory power.

Generally, an MB is decomposed to 4x4 blocks. For each 4x4 block, an area of at most 9x9 pixels is loaded for interpolation. In [8], one 32-bit (4-sample) width RAM is used, so that least 3 cycles are needed to load 9 pixels. Thus, for each 4x4, 27 cycles are required for data loading. Chen et al. [1] propose an interlaced storage format to buffer the AUs in two 64-bit (8-sample) wide RAMs (hereafter as 8Sx2). As shown in Figure 4 (a), by using this 8Sx2 internal memory organization, the required 9 pixels of one row can be fetched in one cycle, which enhances the data throughput. However, this is still not enough for the 4Kx2K applications. As described in Section 2, there are two ways to increase the interpolation throughput. One is vertical expansion, as shown in Figure 4 (b). When using the 8Sx2 scheme with vertical expansion, the memory width is increased proportionally with the data throughput requirement. Even though the memory size is the same, the wider memory width will increase memory area and memory power.

A 4Sx4 (interlaced storage in four 4-sample wide RAMs) scheme is designed to maintain the total memory width while expanding the horizontal parallelism. As described in Figure 4 (c), when the two horizontal neighboring 4x4 blocks have the same MV (or in one partition), at most 13 pixels for each row are required

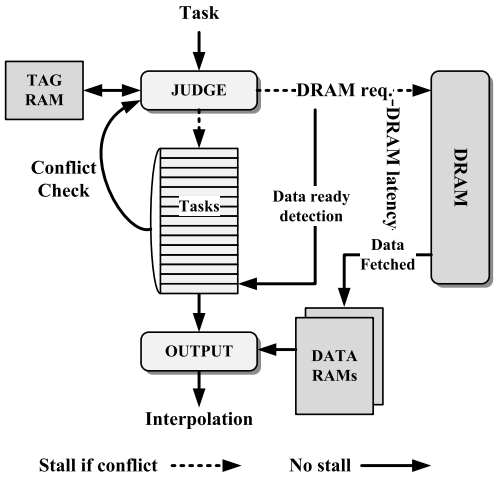

Figure 5: Previous cache architecture.

for interpolation. Four 4-sample wide RAMs with interlaced storage format are applied to ensure generating the 13 samples in one cycle, while maintaining the total memory width to be 16 samples. Based on this 4Sx4 scheme and the interpolator described in Section 2 which processes two rows of luma and chroma samples at same time, the width of each RAM should contain luma and chroma samples of a 4x2 block. The proposed internal memory organization is shown in Figure 3 (c): every AU is divided into four sub-blocks, each of which contains 4x2 luma samples and the corresponding 2x1x2 chroma samples (4:2:0 sampling). These 4 sub-blocks are stored into the 4 different RAMs, while the storing sequence is determined by the lowest bit of parX, for ensuring the neighboring pixels in same two lines are not stored in the same RAM. As a result, each AU can be written to data RAM in one cycle and 32 pixels in 2 lines from different AUs can be read in one cycle.

3.3 Proposed split task queue architecture

In order to tolerate longer memory system latency in the 4Kx2K decoder, Split Task Queue (STQ) architecture is proposed.

Figure 5 shows the previous cache architecture proposed in [1]. Firstly, tasks which describe the location and size of reference block are sent to JUDGE unit which judges miss or hit of AUs inside the reference block according to the TAG RAM. If the needed AUs are not in the data RAM, read requests are sent to DRAM, and then, the fetched AUs are written to data RAM. When all the required data for the task is available in data RAMs and the interpolation unit is ready, the data for this task is output. Because the time from cache sending read requests to receiving the required data from memory system is long, to hide the memory system latency, a task queue is applied after JUDGE unit to store the tasks when waiting the data from memory system. When using the task queue, subsequent basic blocks can be continuously processed during the waiting time. However, in this architecture, conflict checking operation must be processed before JUDGE unit sending current task into the queue to avoid flushing the useful data in the cache. Conflict checking is searching the task queue which stores the previous tasks, and detecting whether the required data of current task will flush the data required by previous tasks. If there is no conflict, the current task is sent to the queue and read requests are sent to DRAM when the AUs needed in this task are not in the data RAM. Otherwise, the JUDGE unit stops sending task to the queue and requests to DRAM, until all conflict tasks are output. Based on this design, the length of the task storing queue is decided by the memory system latency and the speed of interpolation. In the 4Kx2K decoder, the longer system latency will increase the length of the queue. Consequently, the conflict checking operation which checks all the tasks in the queue,

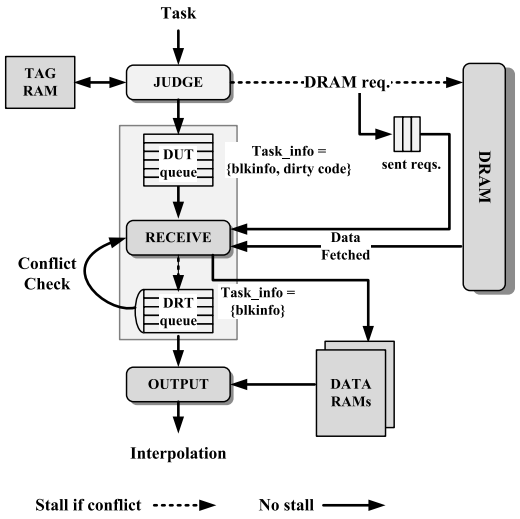

Figure 6: Split Task Queue Architecture.

Table 1: Interpolation throughput.

Sequences[1)]	QP	Inter MB No.	Avg. speed[2)] (cycles/MB)
IntoTrees	24	249236	44.87
	32	211555	36.72
CrowdRun	24	218336	39.66
	32	254880	32.21
ParkJoy	24	156061	35.93
	32	177334	31.17

1): All the sequences are 3840x2160, 10 frames, IBBP.
2): Only considering the processing time of inter MBs.

costs larger gate count. Moreover, longer queue brings higher conflict probability, and results in more idle time.

In order to overcome the above problems, we design to separate the task storing queue into two queues. One stores the data unready tasks called DUT queue, and the other buffers the data ready tasks, called DRT queue, as shown in Figure 6. In the proposed system, JUDGE unit continuously sending tasks to the following DUT queue, and when the required data of the task is available, this task is sent to RECEIVE unit. Then, the RECEIVE unit checks whether the required data of the current data ready task will flush the data required by previous ones stored in DRT queue. If conflict happens, RECEIVE unit stops sending the task to the DRT queue, until all the conflict tasks in DRT queue are output. When the interpolation unit is ready, the task in DRT queue is sent out. Thus, the DRT queue which is utilized for conflict checking can be shorter than the one in previous architecture, since the length of DRT queue is only based on the speed of interpolation. As a result, with the STQ architecture, the influence from longer memory system latency can be reduced, which results in less pipeline stall and lower hardware cost.

4. IMPLEMENTATION RESULTS AND COMPARISON

The proposed architecture is implemented in Verilog HDL on RTL level, and synthesized with Synopsys DesignCompiler by using SMIC 90 G standard cell library. This design is verified both independently in a test environment with inputs given as software generated data, and in a whole Quad-HD video decoder architecture [10].

Table 2: Memory power comparison.

Sequence[1)]	8Sx2 scheme [1][2)] (mW)		Proposed [3)] (mW)		Power Reduction
	Rd.	Wr.	Rd.	Wr.	
IntoTrees	19.56	2.27	10.55	2.77	-38.98%
CrowdRun	15.72	0.80	8.16	0.98	-44.69%
ParkJoy	10.23	0.54	5.29	0.65	-44.79%

1): All the sequences are 3840x2160, 10 frames, QP24,IBBP.
2): Based on 8Sx2, two 384-bit-32-word data RAMs are applied.
3): Based on 4Sx4, four 96-bit-64-word data RAMs are applied.

Table 3: Decoding time comparison.

Sequences[1)]	Without STQ (ms)	With STQ (ms)	Reduction
IntoTrees	258.26	158.46	-38.64%
CrowdRun	260.44	156.96	-39.73%
ParkJoy	190.69	145.85	-23.51%

1): All the sequences are 3840x2160, 10 frames, IBBP, QP24, running @166MHz.

4.1 Interpolation Performance

Table 1 shows the average processing time of interpolation for different sequences, and this value is only for the inter MBs. Due to different MVs and partition sizes, the interpolation processing time for each MB is different. In our work targeting to 4Kx2K application, considering the bi-prediction is not allowed for the partition size smaller than 8x8 on high levels, the worst case is 80 cycles/MB. This case happens when the MB is partitioned to 16 4x4 blocks, each 4x4 block requires a 9x9 block from reference frame, and each 9x9 block is unaligned. Hence, the probability of this case is very low. Moreover, since on level 3 or higher, the maximum MV number of two consecutive MBs should be less than 16, the worst case for one MB which is 80 cycles, only happens when the neighboring MB is intra. So, in this case, the average processing time for the two consecutive MBs is 40 cycles. Considering the maximum MV number limits and Bi-prediction mode is forbidden for the partition size smaller than 8x8, the worst case for two consecutive MBs is 130 cycles. Hence, the worst-case on average processing time for each MB is 65 cycles. The speed requirement in our whole pipelined 4Kx2K decoder is 64 cycles/MB, which is described in [10]. The average speed of the proposed interpolation shown in Table 1, can meet the requirement.

4.2 Cache Memory Features

In order to reduce the internal memory power and area, 4Sx4 scheme is proposed. Based on this internal memory organization, four 96-bit-64-word data RAMs are applied to ensure the interpolation throughput of every cycle two lines with 8 pixels in each line. By using the SMIC register file generator, the memory area of our work is 108800 um^2. The other way to realize a similar throughput with our work, is parallel processing four lines with 4 pixels in each line. For this method, based on 8Sx2 scheme, two 384-bit-32-word data RAMs are required. The memory area of this method is $153836um^2$, which is about 40% larger than ours. Table 2 shows the power comparison between the proposed 4Sx4 based memory organization and 8Sx2 based one. With our memory organization, the data reading power can be reduced by 5∼9mW, since the number of reading times and unit reading power are reduced. Because the same cache size is utilized for these two methods, the total writing data size is the same. Hence, the writing power of our work is a little higher due to the larger memory depth. However, since the unit writing power is lower and the writing ratio is much lower than reading ratio, the total writing power increasing is not significant. Finally, the total memory power reduction can be 39%∼49%.

Table 4: Comparison between this work and state-of-the-art architectures.

	[8]	[9]	[1]	[4], [3][1)]	This work
Max Specification	1920x1080@30fps	1920x1080@30fps	1920x1080@60fps	4096x2160@24fps	3840x2160@60fps
Technology	180nm	180nm	130nm	90nm	90nm
Cache Gate Count	N/A	9k	15.9k [2)]	72k	37.6k
Interpolation Gate Count	20.6k	31k	25.5k	N/A	71.2k
Memory size[3)]	N/A	SP 4kB	TP 4kB	SP 1.5kB	TP 3.1kB
Interpolation Throughput (Worst-case cycles/MB)	560	600	288 (384)[4)]	N/A	65[5)]

[1)]: The cache gate count and max specification are from [4] and [3] , respectively.

[2)]: It is composed of 11k for cache and 4.9k for shifter.

[3)]: SP: single-port SRAM or register file with one R/W port; TP: two-port SRAM or register file with one read port and one write port.

[4)]: Considering the bi-prediction limits on high levels, the throughput is 288 cycls/MB, if not, it is 384 cycles/MB.

[5)]: Considering the maximum MV number and bi-prediction limits on high levels, and worst case on per two consecutive MBs is 130.

4.3 Overall Performance

In the previous cache structure, as shown in Figure 5, the queue which is utilized for conflict checking is 60-bit wide and 20-word deep. The area cost of this queue with conflict checking is 20.8k, when synthesized with Synopsys DesignCompiler by using SMIC 90 G standard cell library. In the proposed STQ architecture as described in Figure 6, the DUT queue is 60-bit wide and 16-word deep, while the DRT queue is 36-bit-wide and 4-word deep. The total area of DUT queue and DRT queue with conflict checking is 15.7k (10.6k for DUT queue and 5.1k for DRT queue with conflict checking). Therefore, the total area can be reduced by 25%.

Beside the low area cost, the STQ architecture can significantly reduce the idle time, which contributes to reducing the overall processing time. Table 3 shows that the decoding time reduction is from 24% to 40%, compared with the architecture without STQ. The InToTree.264 sequence is tested by detail, compared with the architecture without STQ, the cache idle time is reduced by about 90%, and the average processing time is saved by 39%.

4.4 Whole Architecture Performance Comparison

A comparison between this architecture and state-of-the-art works is shown in Table 4. In our design, the worst-case of interpolation throughput is 65 cycles/MB, when considering the maximum MV number and bi-prediction limits on high levels. Compared with the previous works, the throughput is enhanced to at least over 4 times. At the cost of increased parallelism, the logic gate count is also increased. When synthesized with SMIC 90nm process with a timing constraint of 200MHz, the architecture costs a logic gate count of 108.8k including 37.6k for cache and 71.2k for interpolation, which is competitive considering its high performance. Moreover, owing to the 4Sx4 based internal memory organization, the memory area and memory power are optimized. Finally, with the STQ scheme, our design can tolerant longer memory system latency and reduce the decoding time of whole system.

5. CONCLUSION

In this paper, three schemes are proposed to achieve an efficient MC architecture for H.264/AVC real-time decoding of Quad-HD application. Firstly, a high-performance interpolator based on HVE-LCP scheme is proposed to efficiently increase the processing throughput to at least over 4 times as the previous designs. Secondly, an efficient cache memory organization scheme (4Sx4) is adopted to improve the on-chip memory utilization, which contributes to memory area saving and memory power saving of 39%∼49%. Finally, by employing a STQ architecture, the cache system is capable of tolerating much longer latency of the memory system. Consequently, the overall processing time is reduced by 24%∼40%. When implemented with SMIC 90nm process, this design costs a logic gate count and on-chip memory of 108.8k and 3.1kB respectively. We also verified this design both independently in a test environment with inputs given as software generated data, and in a whole Quad-HD video decoder architecture [10].

Acknowledgment

This research was supported by "Ambient SoC Global COE Program of Waseda University" of the MEXT, Japan, and by the JST CREST project.

REFERENCES

[1] X. Chen, P. Liu, D. Zhou, J. Zhu, X. Pan, and S. Goto. A high performance and low bandwidth multi-standard motion compensation design for HD video decoder. *IEICE Trans. Electronics*, E93-C(3):253–260, Mar. 2010.

[2] Y. Chen, C. Cheng, T. Chuang, C. Chen, S. Chien, and L. Chen. Efficient architecture design of motion-compensated temporal filtering/motion compensated prediction engine. *IEEE Trans. CSVT*, 18(1):98 – 109, 2008.

[3] T. Chuang, L. Chang, T. Chiu, Y. Chen, and L. Chen. Bandwidth-efficient cache-based motion compensation architecture with DRAM-friendly data access control. In *Proc. IEEE ICASSP*, pages 2009–2012, 2009.

[4] T. Chuang, P. Tsung, P. Lin, L. Chang, T. Ma, Y. Chen, Y. Chen, C. Tsai, and L.-G. Chen. A 59.5mW scalable/multi-view video decoder chip for quad/3d full hdtv and video streaming applications. In *Dig. Tech. Papers ISSCC*, pages 330 – 331, 2010.

[5] Y. Li, Y. Qu, and Y. He. Memory cache based motion compensation architecture for HDTV H.264/AVC decoder. In *Proc. IEEE ISCAS*, pages 2906–2909, 2007.

[6] C. Lin, J. Chen, H. Chang, Y. Yang, Y. Yang, M. Tsai, J. Guo, and J. Wang. A 160k gates/4.5 KB SRAM H.264 video decoder for HDTV applications. *IEEE JSSC*, 42(1):170 – 182, 2007.

[7] V. Sze, D. Finchelstein, M. Sinangil, and A. Chandrakasan. A 0.7-v 1.8-mw H.264/AVC 720p video decoder. *IEEE JSSC*, 44(11):2943 – 2956, Nov. 2009.

[8] S. Wang, T. Lin, T. Liu, and C. Lee. A new motion compensation design for H.264/AVC decoder. In *Proc. IEEE ISCAS*, pages 4558–4561, 2005.

[9] J. Zheng, W. Gao, and D. Xie. A novel VLSI architecture of motion compensation for multiple standards. *IEEE Trans. Consumer Electronics*, 54(2):687 – 694, May 2008.

[10] D. Zhou, J. Zhou, X. He, J. Kong, J. Zhu, P. Liu, and S. Goto. A 530mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip. In *Dig. Tech. Papers Symp. VLSI Circuits*, pages 171 – 172, 2010.