ABSTRACT

This paper presents a motion compensation (MC) architecture for 8K UHDTV HEVC video decoder. UHDTV’s high resolution significantly increases throughput and memory traffic. Moreover, HEVC supports new coding tools like various sizes of coding unit ranging from 8 to 64. To solve these problems, we propose three optimization schemes. Firstly, four-bank parallel 2D cache organization is proposed to reduce 61.86% memory traffic and support higher interpolator throughput for HEVC. Secondly, we propose pipelined Write-Through mechanism (WTM) to achieve conflict-free performance. Moreover, WTM scheme contributes to around 50% reduction on both memory area and logic gate. Finally, highly parallel interpolator with proposed cache forms integral structure supporting UHDTV. In 90nm process, our design cost 103.6k logic gates with 12kB cache memory. The proposed architecture can support real-time decoding 7680x4320@30fps at 280MHz.

Index Terms— UHDTV, HEVC, motion compensation, interpolation, real-time decoding

1. INTRODUCTION

Ultra High Definition Television (UHDTV) has been popularized in the recent period. By supporting up to 4K or 8K high resolution, UHDTV can provide better visual experience compared to previous Full HD specification. However, higher resolution means to higher data throughput requirement, which leads to several challenges on real-time design. Firstly, massive data volume means that increasing bit rate is required to compress the huge video content. In order to deal with compression problem, new High Efficiency Video Coding has been standardized by JCT-VC in January 2013. As the successor to H.264, HEVC can achieve 50% video compression ratio while guarantee the equivalent compression quality [1]. Secondly, UHDTV increases the design complexity as more hardware resources are needed to ensure higher throughput. Meanwhile, HEVC also adds extra burden on complexity for hardware implementation because of new coding tools introduced for better compression performance, such as flexible hierarchy structure for coding unit (CU) and so on [2]. Thirdly, memory traffic problem becomes more serious for UHDTV. The demand for off-chip memory bandwidth is proportionally increased to the video pixel rate. Bandwidth requirement may exceed more than 10GB/s for UHDTV. In total, new VLSI hardware architecture is required for HEVC-based UHDTV decoder.

Motion compensation is one of the indispensable parts in video decoder. In order to support 8K UHDTV application, MC should be properly designed so that it can solve the high complexity and memory traffic problems mentioned above. In addition, DRAM access latency is another serious issue in design, which may severely degrade the MC performance. Currently Existing motion compensation architectures can’t support the 8K UHDTV application for HEVC standard. New motion compensation architecture is quite desirable for UHDTV.

Several previous motion compensation architectures have been proposed for H.264. For example, in [4] Chen introduced a block-pipelined cache to support real-time decoding 1920x1088@30fps, while Zhou has done further optimization on it and improve the performance to 3840x2160@60fps in [3]. However, these works share one problem that pipelined cache may be hung up because of conflicts in cache writing. For HEVC, Tikekar in [5] designed a whole decoder chip for HEVC draft 4.0. The throughput of this work is 249Mpixels/s, which is 3840x2160@30fps equivalently. Higher throughput is needed for real-time MC design for UHDTV.

In order to achieve real-time decoding for 8K UHDTV, we propose three schemes for motion compensation architecture. Firstly, we propose four-bank parallel 2D architecture to reuse the locality between references and support the high internal throughput with interpolator. By applying this scheme, 61.86% of DRAM access request can be saved. Secondly, we propose the pipelined Write-Through Mechanism (WTM) to equip our MC with conflict-free ability. Compared with previous work, this scheme completely avoids conflict problem to ensure the continuous working of the whole system. Conflict-free ability also reduces the huge circuit overhead for conflict check. Moreover, WTM schemes allow us to replace dual-port (DP) RAM by single-port (SP) one, which can save 43.8% of area.
cost for the data cache. Finally, highly parallel interpolator is introduced to ensure the throughput for UHDTV. Adopted structure supports 8x2 pixels calculated simultaneously for both luma and chroma.

The rest of the paper is organized as follows. Section 2 presents the analysis for MC in HEVC. In Section 3, parallel MC interpolator is discussed. Section 4 shows MC cache design in detail. Hardware implementation results are shown in Section 5. Finally, we conclude the whole paper in Section 6.

2. MC ANALYSIS IN HEVC

2.1. HEVC New Coding Tools for MC

HEVC employs two notable tools in MC compared to H.264. Firstly, it adopts a flexible hierarchical structure for CU and Prediction Unit (PU). HEVC supports CU ranging from 8 to 64 while each CU can be further partitioned into symmetric or asymmetric PUs. Motion compensation is executed on PU level to produce the interpolated results. Secondly, HEVC provides a longer 8-tap filter to realize more accurate interpolation. Depending on fractional motion vector, fractional pixels can be calculated by multiplying neighboring 8 integer or fractional pixels with corresponding filter coefficients. Three types of filters are shown in Table 1. New coding tools enhance compression performance and also result in intensive implementation complexity.

Table 1. Three types of filters

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{-1,4,-10,58,17,-5,1}</td>
</tr>
<tr>
<td>B</td>
<td>{-1,4,-11,40,40,-11,4,-1}</td>
</tr>
<tr>
<td>C</td>
<td>{1,-5,17,58,-10,4,-1}</td>
</tr>
</tbody>
</table>

2.2. Bandwidth Issue in MC

UHDTV application leads to huge demand on bandwidth than ever before. This issue derives from process in which data are fetched from off-chip memory. Taking 8x8 PU as example, 15x15 reference pixels are needed from DRAM. However, assuming the minimum access unit (AU) size between DRAM and cache is 8x8, alignment problem can’t be neglected for real implementation. Inevitably we have to read all pixels in AU even if only some of them are used. The worst case for this alignment problem is shown in Fig. 1. 24x24 reference pixels belonging to 9 AU are fetched for motion compensation process. Around 61% more data overhead are added on DRAM bandwidth compared to actual size of 15x15.

The above discussion is only for one PU. Considering UHDTV, DRAM bandwidth is proportionally increased to the video resolution. In [3], the DRAM bandwidth for 3840x2160@60fps for H.264 is around 7GB/s. Bandwidth requirement will be severely intensive for 8K UHDTV application, which may be larger than 10GBs/frame. How to alleviate the memory traffic is an important issue for motion compensation design.

2.3. Latency and Conflict Issue in MC

Latency problem will degrade the MC performance. Latency is mainly caused by three aspects. Firstly, new specification of DRAM such as DDR3 causes longer latency than before. Secondly, nowadays many new techniques have been introduced in decoder design to enhance the DRAM access efficiency, such as reference recompression [3]. Thus extra cycles for decompression are required. Finally, bus traffic is another important reason producing latency because different process engines share one bus in a whole decoder. To overcome the latency problem, pipelined structure is widely adopted in MC design to hide the latency.

However, conflict problem is further produced by pipelined architecture. As is shown in Fig. 2, pipelined strategy may produce lots of PUs whose reference pixels have already been fetched but not output to interpolator. They will occupy cache and can’t be modified until they are read by interpolator. Conflict happens whenever the coming data from DRAM may flush cache lines which are occupied by an unread PU. In [3][4], huge circuit overhead is maintained to do conflict check. The logic gates for conflict check in [3] are 20.8k, which occupies 55.3% of the whole gates for cache. In [6] a lock bit is added to indicate whether they are occupied or not. However, these works still suffer from the conflict risk which leads to hanging up the pipelined system. Higher definition and frame rate will

![Fig. 1. Alignment Issue: 24x24 Pixels Fetched](image1)

![Fig. 2. Conflict Problem caused by Latency](image2)
further worsen the MC performance. Meanwhile, DP RAM is adopted because we have to support reading and writing cache simultaneously. DP RAM for large cache size takes huge part of the total area. Thus cache architecture that eliminates the effect of both latency and conflict is an urgent need for MC design, especially for UHDTV.

3. PARALLELISM ARCHITECTURE FOR MC INTERPOLATOR

HEVC introduces new 8-tap interpolation algorithm. For UHDTV we have to equip our interpolator with the high throughput ability to support higher resolution videos. Thus, we enhance the parallelism of interpolator in both horizontal and vertical directions as well as luma and chroma.

3.1. Interpolator Filter

Interpolation filter is the elementary component. In [7], adder tree structure for luma interpolator is adopted to replace multiplier. We inherit the adder tree structure and modify the filter coefficients for HM 12.0 like Fig. 3.

3.2. Parallel Implementation for Interpolator

In order to meet the throughput demand for UHDTV, Two parallelism schemes are exploited. Firstly, we construct filters to enhance the horizontal and vertical parallelism. We horizontally expand to 8 pixels in parallel. Additionally, vertical expansion is adopted as Fig. 4. Two adjacent rows are processed simultaneously. Thus, this scheme promotes the interpolator throughput to 8x2 pixels, supporting the demand for UHDTV. Second scheme is luma and chroma parallelism. The motivation comes from the shared motion information for both luma and chroma. With the proposed memory organization in Section 4, luma-chroma parallelism helps reduce DRAM access times.

4. MC CACHE ARCHITECTURE

Motion compensation cache design suffers three challenges. Firstly, the proposed cache should significantly alleviate memory traffic for UHDTV. Secondly, it has to be able to hide latency when accessing DRAM. Finally it should be transparent between DRAM and interpolator. To address these challenges, we propose pipelined four-bank parallel cache architecture with WTM scheme in Fig. 5 to support UHDTV application for HEVC.

4.1. DRAM Organization

Reference pixels are stored in DRAM as the manner of
4.2. Cache Architecture

We propose four-bank parallel 2D cache to reduce the DRAM bandwidth demand as well as realize the seamless joint between system bus and interpolator. Cache is pipelined with Write-Through Mechanism to hide DRAM access latency and completely avoid conflict problem.

4.2.1. 2-D cache Parameter Design

2-D cache is employed to reuse reference data. In order to achieve high reduction of memory traffic with fewer resources, we do several simulations to find out the optimal parameter set. The result is shown in Fig. 6. From the figure we can find that when cache size is larger than 64x64, direct mapping method can achieve almost the same hit-rate performance compared to 2-way set associative mapping. In addition, hardware cost and other factors such as reference recompression are considered together, finally we define cache line as 8x8 and cache size as 64x128 pixels (8x16 cache lines) with direct mapping for each reference list.

4.2.2. Four-Bank Parallel Cache Organization

The cache organization should satisfy two constrains. Firstly, it should be capable to receive the data from system bus. Secondly, as proposed interpolator supports 8x2 pixel throughputs, it implies cache must guarantee 15x2 pixels can be read out from cache per cycle. Thus, we organize our cache as Fig. 7.

The cache consists of 4 banks of SRAM. Each SRAM is 256-word-128-bit size and SP RAM is used to reduce area cost. Thus the cache line size is 8x2 luma pixels, which is exactly equal to the width of system bus. Chroma part inherits the same organization. Considering interface with DRAM, as we have defined the AU size is 8x8 pixels, it means 6 cache lines should be replaced if cache miss happened. For interface with interpolator, four banks of SRAM can provide at most 32x2 pixels (four cache lines) per cycle to the interpolator. As the maximum input requirement for interpolator is 15x2 in our design, which may locate in three cache lines in worst case, our proposed organization can completely ensure the internal bandwidth. Thus our proposed organization can act as a transparent role between DRAM and interpolator.

4.2.3. Pipelined Write-Through Mechanism

The whole architecture is shown in Fig. 5. Process blocks are pipelined so that subsequent operations can be done while waiting for the DRAM latency. Thus the conflict problem mentioned in Section 2.3 will happen. Write-Through Mechanism is the proposed hardware scheme to realize conflict-free cache writing process.

In previous works, the reason for conflict is that fetched AU is written into cache immediately even if it will be read by interpolator several cycles later. In this case, these AUs occupy the cache and we can’t flush them until they are output. On this basis, we can avoid conflict if we change the strategy so that AU is written into cache only when it is needed by interpolator. Based on this idea, Write-Through Mechanism hardware architecture is proposed in Fig. 8(a).

In order to support the WTM scheme, we maintain a Fetched AU FIFO ahead of each bank of cache data RAM. The data path for WTM is like this. At first, each PU is split into strips with the fixed width equal to 8 by Task Producer in Fig. 5. We call each strip as task, whose width is fixed as eight so that it can properly fit for the interpolator interface. Based on the motion information, reference region can be obtained for each task. Then the corresponding AUs are checked whether they are hit or miss. Hit/Miss information is pushed into Status FIFO for coming use. AU is then stored into corresponding FIFO based on the last two index bit.
Step 1: Read

Data Cache

RAM_0 W/R
RAM_0 rDATA
RAM_1 W/R
RAM_1 rDATA
RAM_2 W/R
RAM_2 rDATA
RAM_3 W/R
RAM_3 rDATA

Status FIFO_0
Status FIFO_1
Status FIFO_2
Status FIFO_3

HIT
HIT
HIT
MISS

Read
Bank 0

Bank 1

Bank 2

Bank 3

8x2
8x2
8x2
8x2

(a) WTM Hardware Structure. Four-bank parallel cache is utilized and each bank has a corresponding Fetched AU FIFO & Status FIFO.

(b) Timing for WTM. Considering 15x2 pixels locates in three continuous cache lines (RAM_0/1/3), and cache lines is hit for RAM_0 & RAM_1 while missed for RAM_3.

Fig. 8. Working Mechanism for WTM

This work is done in Buffer Switcher module. Output-Controller module will monitor the completeness of reference fetching for each task. If so, reference data will be output to interpolator for calculation.

In detail, WTM-based cache works in serial-in-parallel-out mechanism to eliminate throughput difference between system bus and interpolator while supporting conflict-free. On system bus side, each fetched AU is pushed into corresponding FIFO serially in six cycles. On interpolator side, Output Controller is in charge of reading at most 15x2 pixels from cache and output them to interpolator. Firstly, Status FIFO is checked for each AU. If AU is hit, data can be read out from cache directly. If AU is missed, it means AUs are fetched from DRAM and currently saved in corresponding Fetched AU FIFO. Thus WTM scheme will parallelly read data from FIFO and write them into cache. As proposed cache utilizes the write through mode, the written data will be available at the read data ports next cycle. By scheduling timing in Fig. 8(b), the data can be read out from four banks of RAM synchronously no matter whether they are saved in FIFO or cache. Noticing WTM scheme promises each bank can only be either write or read at any clock edge, it allows us to construct the cache by SP RAM instead of DP RAM. Based on WTM, any data written into cache is output to interpolator immediately so it can be flushed by subsequent data. Thus WTM based cache avoids the conflict risk to improve the pipelined cache performance.

5. IMPLEMENTATION RESULT

We implement our proposed motion compensation architecture on RTL level design using Verilog HDL and further synthesize it by Synopsys Design Compiler with TSMC 90nm process. Both Lowdelay and RandomAccess configurations are used for verifying the correctness of our design in different test environment.

5.1. Performance

Firstly, we adopt highly parallel interpolator for UHDTV. The throughput can achieve 8x2 pixels per cycle after filling the pipeline. Considering the worst case shown in Fig. 9, all pixels in frames are inter-predicted with bi-prediction and all the partition size is nLx2N or nRx2N with CU equal to 16x16. Though the worst case is so strict that it has little possibility to be occurred, our proposed interpolator can achieve 3.56pixels/cycle, meeting the requirement for 8K UHDTV under 300MHz clock frequency.

Secondly, bandwidth demand is an important measure criterion for MC. WTM cache helps reduce 61.86% bandwidth reduction for RandomAccess while 65.56% for Lowdelay. If no cache adopted, DRAM bandwidth demand for 7680x4320@30fps with bi-prediction is 9.0GB/s, while proposed MC cache can significantly reduce it to 3.4GB/s. Meanwhile, the proposed MC cache is conflict-free cache because of WTM scheme. WTM helps eliminate the idle cycle of cache, which is 20ms for decoding one frame in [4].

![Fig. 9. Worst case for interpolator](image)

![Fig. 10. Logic Gate vs. Timing constrain](image)
This paper proposes a HEVC motion compensation architecture for UHDTV. The design can accomplish the new coding tools in HEVC such as various sizes of coding unit and new interpolation algorithm. In particular, we introduce four-bank parallel cache organization reduces the memory traffic by 61.86%. Moreover, WTM-based cache promises conflict-free ability which reduce more than half of the logic gate overhead for cache, and support forming data cache using SP RAM instead of DP RAM to reduce 43.8% of memory area. In total, the proposed MC supports 7680x4320@30 fps real-time video decoding at 280MHz with no conflict and less area cost.

### 5.2. Synthesis Result

The MC architecture contains memory part and two logic parts. For logic parts interpolator and cache controller are synthesized. Interpolator takes major part of the area cost which cost 64.7k gates for luma and 25.6k for chroma at the 250MHz constrain while the controller part consists of only 13.3k gates. In Fig. 10, we show the total gate in different time constraints compared [6].

For MC memory, we use four banks of SP RAM to organize data RAM, each of which is 256-word-128-bit. The total size is 8kB for luma. The chroma part is organized in the same manner. Synthesized by TSMC 90nm memory generator, the area of proposed data cache is 59948.0μm² for each bank. It benefits from the WTM scheme which allows use SP RAM to save area cost. Assuming no WTM is used, the area is 106664.6μm² for DP RAM, which is around twice larger than ours. The result shows our WTM based MC cache architecture saves memory area by 43.8%.

### 5.3. Comparison with Other Works

Table 2 shows the comparison with other related works. Most of previous works are aiming at H.264. We can observe that our proposed MC architecture can support new HEVC standard without increasing much hardware resource. Meanwhile, the logic gates in [3] for conflict check are 20.8k while our proposed WTM cache realizes conflict-free performance. Thus we can reduce the conflict-check circuit to achieve only 13.3k gates for cache compared to 37.6k in [3]. In [6] the decoder chip for HEVC is designed. Compared with it, our design improves the performance as high as 4 times with fewer resources and memory area. Totally, our proposed motion compensation architecture can real-time decode 7680x4320@30fps video at 280MHz.

### 6. CONCLUSION

Table 2. Comparison with state-of-the-art works

<table>
<thead>
<tr>
<th></th>
<th>[4]</th>
<th>[6]</th>
<th>[3]</th>
<th>[5]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>H.264</td>
<td>H.264</td>
<td>H.264</td>
<td>HEVC (HM4.0)</td>
<td></td>
</tr>
<tr>
<td>Max. Pixel Rate</td>
<td>63MPixel/s</td>
<td>213MPixel/s</td>
<td>498MPixel/s</td>
<td>249MPixel/s</td>
<td></td>
</tr>
<tr>
<td>Max. Resolution</td>
<td>1920x1080@60fps</td>
<td>4096x2160@24fps</td>
<td>3840x2160@60fps</td>
<td>3840x2160@30fps</td>
<td></td>
</tr>
<tr>
<td>Gate Interpolator</td>
<td>15.9k</td>
<td>72k</td>
<td>37.6k</td>
<td>126k*</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td>130nm</td>
<td>90nm</td>
<td>90nm</td>
<td>40nm</td>
<td></td>
</tr>
<tr>
<td>Total Memory Size</td>
<td>4kB (TP 3kB)</td>
<td>N/A (SP 1.5kB)</td>
<td>N/A (TP 3.1kB)</td>
<td>19.4kB (SP 12kB)</td>
<td></td>
</tr>
</tbody>
</table>

*: The gate count is only for MC Cache module, and other MC parts are packed in Prediction module. Thus, actual gates for MC may be larger than this.

### 7. REFERENCES


