Waseda University Doctoral Dissertation

Research on Inter Prediction Algorithms and Architectures for High-Performance Video Codec VLSI

Jinjia ZHOU

Graduate School of Information, Production and Systems
Waseda University

February 2013
Abstract

While 1080 HD is the current standard of mainstream video applications such as TV broadcasting, even higher specifications such as 4K Ultra HD have been targeted by next-generation applications. To store and transmit these mass video contents, video compression is indispensable. From MPEG-1/2/4 to H.264/AVC and HEVC etc., the continuous innovation in this area has been a significant stimulation of the popularization of multimedia in modern life. In almost all these compression standards, inter prediction is one of the most important coding tools, which significantly contributes to coding efficiency by exploring the temporal data redundancy between neighboring frames. In the meanwhile, inter prediction also involves high complexity and huge memory bandwidth. In the video encoder and decoder, inter prediction is realized by motion estimation (ME) and motion compensation (MC), respectively.

ME searches neighboring reference frames to find the pixel blocks which best match the blocks in the current frame. As a result, only the blocks’ differences along with a set of displacements called motion vector (MV) are required to encode the current frame. Most of the hardware ME architectures, especially those implemented in recently published video encoder chips (Y.K. Lin; ISSCC2008, L.F. Dong; ISSCC2009) are based on full search or modified versions of full search. In order to find the best matching block, full search ME checks all points in the search area which leads to huge computational complexity. In this dissertation, we present the alternating asymmetric search range assignment (AASRA) schemes including
AASRA-B, AASRA-P and AASRA-PB to reduce the complexity of full search ME while maintaining coding performance.

MC utilizes MVs to locate the matching blocks, and then compensates the blocks’ differences to decode the current blocks. VLSI design for MC is challenged by the computational complexity of fractional pixel interpolation, the high memory bandwidth of reference frame retrieval and the long latency of external memory systems. Many works (D. Zhou; ISCAS2007, S. Wang; ISCAS2005, V. Sze; JSSC2009) have been done to reduce the complexity of parallelized interpolation, and many architectures have been designed (X. Chen; IEICE2009, T. Chuang; ICASSP2009) to reduce the memory bandwidth. In this dissertation, three algorithms are proposed to obtain more efficient interpolation and less on-chip memory bandwidth than the previous works.

In latest standards, MVs are also predicted and compressed for bit rate saving. Hence MV decoding becomes an important component of the video decoder to restore the current MVs from adjacent MVs and MV differences. This also involves considerable memory bandwidth. Moreover, due to the flexible block size of inter predication, the control complexity of MV decoding is critical. Most of the previous works on MV calculation architectures get variable processing time for each MB, and it results in high control complexity (K. Yoo; ICIP2008, H. Yin; ICALIP2008). In order to decrease the control complexity, a dual-mode based stable 64 cycles/MB pipeline is proposed. Moreover, the strategy for reducing the memory bandwidth is also proposed in Chapter 3.

In this dissertation, efficient algorithms and architectures are proposed to reduce the computational complexity and memory bandwidth of inter prediction,
and consequently decrease the area cost and memory power consumption of video codec VLSI.

This dissertation consists of the following 6 chapters.

**Chapter 1 [Introduction]** introduces the background knowledge of video compression, and the main challenges for inter prediction. The main contributions and an overview of this dissertation are also presented in this chapter.

**Chapter 2 [High-Throughput Motion Compensation]** presents a 16-65 cycles/MB high throughput motion compensation (MC) architecture.

1) A high-performance interpolator parallelizes the horizontal and vertical filtering to efficiently increase the processing throughput to at least over 4 times as the previous designs of S. Wang (ISCAS2005) and D. Zhou (ISCAS2007). When comparing with vertical expansion architecture of V. Sze (JSSC2009), this work also increases the throughout to 2 times.

2) An efficient cache memory organization scheme (4Sx4) splits the memory width and stores the data in an interlaced way to improve the on-chip memory utilization. As a result, it contributes memory area saving of 25% and memory power saving of 39%~49%, when comparing with the X. Chen’s work (IEICE2009).

3) A Split Task Queue (STQ) architecture separates the task storing queues into two stages of the pipeline to hide the memory latency and reduce the pipeline stall. Consequently, the cache idle time is saved by 90%, which contributes to reducing the overall processing time by 24%~40%.

Experimental results show this design is capable of real-time 4kx2k@60fps decoding at 166MHz, with 108.8k logic gates and 3.1kB on-chip memory.
Comparing with current 4kx2k@24fps decoder chip (T. Chuang; ISSCC2010), this work increases the throughput to 2.5 times.

**Chapter 3 [Efficient Joint Parameter Decoder]** presents a joint parameter decoder to realize the calculation of motion vector, intra prediction mode and boundary strength.

1) Dual-mode pipeline scheme categorizes the various partition sizes and prediction modes into two control modes. And then, the pipeline is derived by the simplified two modes to increase system throughout and reduce the control complexity. As a result, a constant-throughput of 64cycles/MB is obtained. The number of clock cycles required for processing one MB is reduced by 75% from state-of-the-art works (K. Yoo; ICIP2008, H. Yin; ICALIP2008).

2) Three-step bandwidth reduction strategy is proposed to condense the data. On step 1, a partition based storage format is applied to condense the MB level data. On step 2, variable length coding based compression method is utilized to reduce the data size in each partition. Finally, the total bandwidth is further reduced by combining the co-located and last-line information. Consequently, 85%~98% bandwidth saving is achieved.

Experimental results show this design is capable of real-time 4kx2k@60fps decoding at 166MHz, with 37.2k logic gates. Comparing with K. Yoo (ICIP2008) and H. Yin (ICALIP2008), the throughput of our design is increased from 260 cycles/MB to 64 cycles/MB, with smaller logic gates.

**Chapter 4 [Alternating Asymmetric Search Range Assignment for Motion Estimation]** presents alternating asymmetric search range assignment (AASRA) schemes for motion estimation.
1) AASRA-B uses a large and a small search ranges, respectively, for the two reference directions of bidirectional ME. The assignment of these two search ranges alternates between past and future references for each MB/CTB, enabling ME in both directions to track high motions. AASRA-P extends the application of AASRA to P, and AASRA-PB combines the features of AASRA-B and AASRA-P. The three schemes can reduce ME complexity by 46%, 46% and 70%, respectively, with small coding performance drop.

2) The proposed AASRA schemes also have the adaptability to be combined with many existing fast algorithms and architectures to achieve an incremental reduction of complexity. Necessary adaptations of AASRA are proposed to combine with the existing works. When combining Y. Lin’s (TCSVT2008) design with AASRA, the hardware cost can be reduced by 33%. When implemented the architecture of C. Kao (TVLSI2010) with AASRA, the throughput can be increased by 43%~64%.

Chapter 5 [Architectures Implemented in Video Decoder Chip] summaries the features of inter prediction related components in the implemented the video decoder chips. The proposals in Chapter 2 and Chapter 3 have been silicon proven in two decoder chips.

Chapter 6 [Conclusion] concludes the contributions of this dissertation.
Contents

Abstract ........................................................................................................................................... i

Contents ........................................................................................................................................ vi

List of Tables ................................................................................................................................. x

List of Figures ................................................................................................................................. xi

1 Introduction .................................................................................................................................. 1

1.1 Background ............................................................................................................................... 1

1.2 Contributions of the Dissertation ............................................................................................. 3

1.3 Organization of the Dissertation ............................................................................................... 5

2 High-Throughput Motion Compensation ....................................................................................... 6

2.1 Design Challenges ....................................................................................................................... 7

2.2 Parallelism of MC Interpolation ................................................................................................. 9

2.2.1 MC Interpolation .................................................................................................................... 9

2.2.2 Parallelism Analysis ............................................................................................................. 11

2.2.3 Horizontal-Vertical Expansion and Luma-Chroma Parallelism ............................................ 13

2.3 Cache Architecture .................................................................................................................... 15

2.3.1 Cache System Design .......................................................................................................... 16

2.3.2 2-D Cache Mapping ........................................................................................................... 17
3.5 Extended from H.264/AVC to HEVC ......................................................... 58

3.6 Implementation Results and Comparison ......................................................... 59
  3.6.1 Throughput Comparison .............................................................................. 59
  3.6.2 DRAM Bandwidth Comparison .................................................................. 60
  3.6.3 Hardware Cost Comparison ....................................................................... 62
  3.6.4 Function Comparison ................................................................................ 62

3.7 Summary ........................................................................................................... 63

4 Alternating Asymmetric Search Range Assignment for Motion Estimation .... 65
  4.1 Recent Works on Motion Estimation .............................................................. 66
  4.2 Alternating Asymmetric SR Assignment Algorithms ..................................... 67
    4.2.1 AASRA for Bidirectional ME (AASRA-B) ................................................. 67
    4.2.2 Applying AASRA in P Frames (AASRA-P) .............................................. 72
    4.2.3 Combining AASRA-P and AASRA-B (AASRA-PB) ............................... 73
    4.2.4 Hardware Complexity Analysis ................................................................ 75
  4.3 Combined with Existing ME Architectures ...................................................... 79
    4.3.1 AASRA Combined with MB-parallel ME .................................................. 80
    4.3.2 AASRA Combined with Hierarchical ME ............................................... 82
  4.4 Experiments and Evaluation .......................................................................... 83
    4.4.1 Equivalent Fixed Search Range for AASRA .......................................... 83
    4.4.2 Rate-Distortion Based Evaluation ............................................................. 88
  4.5 Summary ......................................................................................................... 95

5 Architectures Implemented in Video Decoder Chip ........................................... 96
5.1  Implemented Chips ................................................................. 96

5.2  Features of Inter Prediction Components in Decoder Chip .................. 98

6  Conclusion .............................................................................. 101

Acknowledgement ...................................................................... 103

References .................................................................................. 104

Publications ................................................................................ 113
List of Tables

Table 2-1 Interpolation throughput .................................................................28
Table 2-2 Memory power comparison ...............................................................31
Table 2-3 Decoding time comparison .................................................................32
Table 2-4 Comparison between this work and state-of-the-art MC architectures ...34
Table 3-1 Bandwidth profiling result .................................................................40
Table 3-2 Control Mode Categorization ............................................................46
Table 3-3 Signed Exp-Colomb code .................................................................54
Table 3-4 Unary code .........................................................................................55
Table 3-5 Comparison between this work and state-of-the-art MD architectures ...59
Table 3-6 Bandwidth test results on various sequences ......................................61
Table 3-7 Gate count summary ..........................................................................62
Table 4-1 Complexity comparison by EFSR ......................................................87
Table 4-2 Test conditions ..................................................................................88
Table 4-3 RD performance of AASRA on full-search JM. ..................................91
Table 4-4 RD Performance of different SR.L-SR.S ratio. .................................92
Table 4-5 RD performance of AASRA on full-search HM ..................................93
Table 4-6 RD Performance of AASRA on IMNPDR and PMRME ....................94
Table 5-1 Pipeline utilization of major components [33]....................................98
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2-1</td>
<td>4-pixel parallel interpolation architecture</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2-2</td>
<td>Horizontal expansion</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2-3</td>
<td>Vertical expansion</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2-4</td>
<td>Data alignment problem</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2-5</td>
<td>Horizontal-Vertical Expansion</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2-6</td>
<td>Cache mapping and internal memory organization analysis</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2-7</td>
<td>Cache memory design</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2-8</td>
<td>8Sx2 internal memory organization</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2-9</td>
<td>8Sx2 internal memory organization with vertical expansion</td>
<td>20</td>
</tr>
<tr>
<td>Figure 2-10</td>
<td>4Sx4 internal memory organization with horizontal expansion</td>
<td>21</td>
</tr>
<tr>
<td>Figure 2-11</td>
<td>Cache architecture</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2-12</td>
<td>Cache architecture with STQ</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2-13</td>
<td>Motion compensation system architecture</td>
<td>25</td>
</tr>
<tr>
<td>Figure 2-14</td>
<td>4-pixel based interpolation in HEVC</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2-15</td>
<td>8-pixel based interpolation in HEVC</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2-16</td>
<td>Example of worst case per two consecutive MBs</td>
<td>29</td>
</tr>
<tr>
<td>Figure 2-17</td>
<td>Interpolation throughput analysis</td>
<td>29</td>
</tr>
<tr>
<td>Figure 2-18</td>
<td>Cache efficiency comparison</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3-1</td>
<td>Pipeline architecture analysis</td>
<td>42</td>
</tr>
<tr>
<td>Figure 3-2</td>
<td>Pipeline schedule</td>
<td>46</td>
</tr>
<tr>
<td>Figure 3-3</td>
<td>Determination of the neighboring blocks</td>
<td>47</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td>JPDEC pipeline mechanism for different control modes</td>
<td></td>
</tr>
<tr>
<td>3-5</td>
<td>Partition based storage and DPCM based variable length coding</td>
<td></td>
</tr>
<tr>
<td>3-6</td>
<td>Probability Distribution of MV DPCM and Exp-Golomb Code</td>
<td></td>
</tr>
<tr>
<td>3-7</td>
<td>Probability distribution of refIdx and unary code</td>
<td></td>
</tr>
<tr>
<td>3-8</td>
<td>Integrated Coding for Co-located and Last-line Data</td>
<td></td>
</tr>
<tr>
<td>3-9</td>
<td>Connection between DRAM and pipelined calculation part</td>
<td></td>
</tr>
<tr>
<td>4-1</td>
<td>Alternating asymmetric search range assignment for B frames</td>
<td></td>
</tr>
<tr>
<td>4-2</td>
<td>Comparison on motion vector tracking capabilities</td>
<td></td>
</tr>
<tr>
<td>4-3</td>
<td>Motion vector information encoded by SR.L-SR.S=16-8</td>
<td></td>
</tr>
<tr>
<td>4-4</td>
<td>Comparison between AASRA-B and original full search</td>
<td></td>
</tr>
<tr>
<td>4-5</td>
<td>AASRA for P frames</td>
<td></td>
</tr>
<tr>
<td>4-6</td>
<td>SR.L rotation of AASRA-B</td>
<td></td>
</tr>
<tr>
<td>4-7</td>
<td>Memory access sequence of snake scan</td>
<td></td>
</tr>
<tr>
<td>4-8</td>
<td>Additional control logic required to implement AASRA</td>
<td></td>
</tr>
<tr>
<td>4-9</td>
<td>Typical organization of ME related components in video encoder</td>
<td></td>
</tr>
<tr>
<td>4-10</td>
<td>The SR.S window is always contained by the SR.L one</td>
<td></td>
</tr>
<tr>
<td>4-11</td>
<td>Search center decision for AASRA based IMNPDR</td>
<td></td>
</tr>
<tr>
<td>4-12</td>
<td>Relative hardware parallelism requirement comparison</td>
<td></td>
</tr>
<tr>
<td>4-13</td>
<td>Evaluation of equivalent fixed search range</td>
<td></td>
</tr>
<tr>
<td>4-14</td>
<td>Evaluation of EFSR on the 1080p Pedestrian sequence</td>
<td></td>
</tr>
<tr>
<td>4-16</td>
<td>Subjective quality comparison</td>
<td></td>
</tr>
<tr>
<td>5-1</td>
<td>Chip micrograph of 8kx4k video decoder chip [58]</td>
<td></td>
</tr>
<tr>
<td>5-2</td>
<td>Chip micrograph of 4kx2k video decoder [15]</td>
<td></td>
</tr>
<tr>
<td>5-3</td>
<td>Top-level block-diagram of 4kx2k@60fps video decoder [33]</td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Background

While 1080 HD is the current standard of mainstream video applications such as TV broadcasting, even higher specifications such as 4K Ultra HD have been targeted by next-generation applications. To store and transmit these mass video contents, video compression is indispensable. From MPEG-1/2/4 [1][2][3] to H.264/AVC [4][5] and HEVC [6] etc., the continuous innovation in this area has been a significant stimulation of the popularization of multimedia in modern life. In almost all these compression standards, inter prediction is one of the most important coding tools, which significantly contributes to coding efficiency by exploring the temporal data redundancy between neighboring frames. In the meanwhile, inter prediction also involves high complexity and huge memory bandwidth. In the video encoder and decoder, inter prediction is realized by motion estimation (ME) and motion compensation (MC), respectively. In latest standards, MVs are also predicted and compressed for bit rate saving. Hence MV decoding becomes an important component of the video decoder to restore the current MVs from adjacent MVs and MV differences.

With the increasingly higher resolutions applications such as 1080p HD, 4K and 8K Ultra HD (or Super Hi-Vision, SHV), inter prediction is challenged by the following aspects.

On one hand, with higher specifications, the complexity of inter prediction is increased significantly. Firstly, in order to support the precision of half and quarter
pixels, interpolation which is one of the most computation intensive component, is required in both ME and MC components. To increase the throughput, computational complexity is further increased for the parallelized interpolation. Secondly, due to the use of variable-block-size inter prediction and multiple prediction modes, MD decoding which is one of the most algorithm-irregular components of the H.264/AVC frame work, requires high complexity controller. Finally, in order to find the best matching block, full-search ME checks all points in a large search area and consequently it involves huge computational complexity.

On the other hand, inter-frame coding is challenged by the huge memory bandwidth which brings large power consumption. Firstly, both ME and MC require to fetch back the pixels of neighboring reference frames stored in off-chip DRAM. The reference read operation composes a dominant portion of DRAM traffic for a whole video decoding or encoding system. Secondly, in general, data buffer is utilized to reduce the DRAM bandwidth from reference read operation. However, sending the data from buffer to calculation component brings large on-chip memory bandwidth requirement, which should be seriously considered. Finally, in MD decoding, motion information for a whole co-located picture (coIPic) and the last decoded line (picture width) is required, and consequently, mass data must be stored and fetched back from DRAM which leads to large amount of external memory accesses.

In order to solve the high complexity problems, various techniques are proposed. Firstly, Horizontal-Vertical Expansion and Luma-Chroma Parallelism (HVE-LCP) algorithm is proposed to reduce the computational complexity by efficiently parallelizing the interpolation component. Secondly, a 64-cycle-per-MB pipeline with simplified control modes is designed to reduce the complexity of MD decoding controller. Finally, to reduce the complexity of full-search ME which is determined by
the search range, the alternating asymmetric SR assignment (AASRA) schemes are proposed.

In order to solve the huge memory bandwidth problems, the following schemes are proposed. Firstly, the cache system is adopted to reduce the DRAM traffic from reference frame read operation. Secondly, in order reduce the on-chip bandwidth between cache and calculation component, an efficient cache memory organization scheme (4Sx4) is applied. Finally, three-step bandwidth reduction strategy is proposed to save memory bandwidth of motion data decoding components.

1.2 Contributions of the Dissertation

In this dissertation, efficient algorithms and architectures are proposed to reduce the computational complexity and memory bandwidth of inter prediction, and consequently decrease the area cost and memory power consumption of video codec VLSI.

Firstly, cache based motion compensation (MC) architecture is proposed to support real-time 4kx2k@60fps decoding at less than 200MHz, with 37.2k logic gates. A **high-performance interpolator** based on horizontal-vertical expansion and luma-chroma parallelism (HVE-LCP) is proposed to efficiently increase the processing throughput to at least over 4 times as the previous designs. Then, an **efficient cache memory organization scheme (4Sx4)** is adopted to improve the on-chip memory utilization, which contributes to memory area saving of 25% and memory power saving of 39%~49%. By employing a **Split Task Queue (STQ) architecture**, the cache system is 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%.
Secondly, VLSI architecture of a joint parameter decoder is proposed to realize the calculation of motion vector (MV), intra prediction mode (IPM) and boundary strength (BS). For this architecture, a dual-mode pipeline with simplified control modes is designed to increase system throughput and reduce hardware cost. Then, three-step bandwidth reduction strategy is applied. The data which includes the motion information for the co-located picture and the last decoded line, is pre-processed before being stored to DRAM. A partition based storage format is applied to con- dense the MB level data, while variable length coding based compression method is utilized to reduce the data size in each partition. Experimental results show our design is capable of real-time 3840x2160@60fps decoding at less than 166MHz, with 37.2k logic gates. Meanwhile, by applying the proposed scheme, 85%~98% bandwidth saving is achieved, compared with storing the original information for every 4x4 block to DRAM.

Finally, alternating asymmetric search range assignment (AASRA) schemes for motion estimation is proposed which includes AASRA-B, AASRA-P and AASRA-PB. AASRA-B uses a large and a small search ranges, respectively, for the two reference directions of bidirectional ME. The assignment of these two search ranges alternates between past and future references for each MB/CTB, enabling ME in both directions to track high motions with reduced complexity. AASRA-P extends the application of AASRA to P frames, and AASRA-PB combines the features of AASRA-B and AASRA-P to obtain even higher complexity saving. AASRA-B, AASRA-P and AASRA-PB reduce ME complexity by 46%, 46% and 70%, respectively, with small coding performance drop. Then, we propose an equivalent fixed search range (EFSR) metric to compensate the coding efficiency degradation in the complexity comparison. After the compensation, AASRA still shows 28.9% to 43.1% complexity reduction. We also demonstrate AASRA’s flexibility to be
combined with the state-of-the-art ME architectures including MB-parallel ME and hierarchical ME.

The cache based motion compensation is implemented in the 2Gpixels/s H.264/AVC HP/MVC video decoder chip and 530Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip. The joint parameter decoder is implemented in 2Gpixels/s H.264/AVC HP/MVC video decoder chip, 530Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip, and 1080p@60fps multi-standard video decoder chip.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. Chapter 2 discusses the cache based motion compensation architecture including a high-performance interpolator, an efficient cache memory organization scheme (4Sx4), and a Split Task Queue (STQ) architecture. Chapter 3 presents VLSI architecture of a joint parameter decoder is proposed to realize the calculation of motion vector (MV), intra prediction mode (IPM) and boundary strength (BS). This architecture is optimized by a dual-mode pipeline and three-step bandwidth reduction strategy. In Chapter 4, alternating asymmetric search range assignment (AASRA) schemes for motion estimation is proposed which includes AASRA-B, AASRA-P and AASRA-PB. Chapter 5 summaries the features of inter prediction related components in the implemented the video decoder chip. Chapter 6 concludes the contributions of this dissertation.
2 High-Throughput Motion Compensation

In this chapter, a 16-65 cycles/MB high-throughput motion compensation (MC) architecture for 4kx2k H.264/AVC video decoder is presented.

In section 2.1, the main design challenges are presented including the huge area cost and power consumption. Moreover, the long memory system latency leads to performance drop of the MC pipeline.

In section 2.2, a high-performance interpolator based on Horizontal-Vertical Expansion and Luma-Chroma Parallelism (HVE-LCP) is proposed to efficiently increase the processing throughput to at least over 4 times as the previous designs.

In section 2.3, an efficient cache memory organization scheme (4Sx4) is adopted to improve the on-chip memory utilization, which contributes to memory area saving of 25% and memory power saving of 39%~49%. Moreover, by employing Split Task Queue (STQ) architecture, the cache system is capable of tolerating much longer latency of the memory system.

In section 2.4, the overall MC architecture is summarized. The techniques about using request queue and dirty code are illustrated.

Finally, section 2.6 gives implementation results and comparison. When implemented with SMIC 90nm process, this design costs a logic gate count and on-chip memory of 108.8k and 3.1kB respectively. The proposed MC architecture can support real-time processing of 3840x2160@60fps with less than 166MHz.
2.1 Design Challenges

For real-time decoding 3840x2160@60fps, the required throughput should be 500Mpixels/s. With current technologies, H.264/AVC high-profile decoding for 1080p@60fps requires a DRAM configuration of 32-bit DDR-400 (1.6GB/s) [7]. Therefore, for 3840x2160@60fps, the bandwidth requirement increases by at least 4.3 times proportionally to the throughput, to be near 7GB/s. For the cache and interpolation components, the computational power also increases by 4.3 times when compared with the architecture applied for 1080@60fps decoder. Assuming the whole LSI system is working at 166MHz, the theoretical time budget for real-time processing a 3840x2160@60fps sequence should be no more than 88 cycles for each component. In practice, it is better to be working at less than 70cycles per MB so as to preserve enough space for the system. The 4kx2k motion compensation(MC), which is speed bottleneck of the whole decoder, are 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. In order to meet this requirement, the straightforward way is parallelizing the processing unit from the previous one row [8][9] to four rows. Although the parallelized architecture can increase the throughput, the critical data alignment problem will lead to extra overhead on both the memory read power and interpolation processing time. This means the cost for parallelism will be larger than the enhancement in throughput.

Secondly, with higher specifications, memory bandwidth requirement increases significantly. Previous contributions proved that the cache systems [10] [11] [12] [13] [14] can be an effective way to reduce the external DRAM bandwidth. However, the on-chip memory bandwidth from the cache system to the interpolation component
High-Throughput Motion Compensation

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 since higher-speed DRAM specifications such as DDR2 and DDR3. The CAS latency in terms of absolute time is almost the same among DRAM generations, but with the clock frequency increased, the CAS latency in terms of number of clock cycles increases, which influences the cache architecture. On the other hand, new techniques adopted to enhance the DRAM access efficiency, such as reference frame recompression [15], 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, in order 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 chapter 2. 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.

In order to solve the above problems, three schemes are proposed in this paper to achieve an efficient MC architecture for H.264/AVC real-time decoding of 4kx2k applications. 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 power is saved by 39–49%, and memory area is reduced by 25%.
Finally, by employing a Split Task Queue (STQ) architecture, the cache system is 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%.

### 2.2 Parallelism of MC Interpolation

Compared with HD application, the throughput requirement for MC interpolation in Quad-HD cases is increased by at least 4 times. In order to meet this requirement, the straightforward way is parallelizing the processing unit from the previous one row [8] [9] to four rows. Although the parallelized architecture can increase the throughput, the critical data alignment problem will lead to extra overhead on both the memory read power and interpolation processing time. This means the cost for parallelism will be larger than the enhancement in throughput.

In order to solve the above problems, 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.

#### 2.2.1 MC Interpolation

Most of the previous works on MC interpolation decompose an MB 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. This 4-pixel parallel luma interpolater architecture is shown in Figure 2-1.
Luma interpolation involves using a 1-D 6-tap filter to generate the half-pel locations. A 1-D bilinear filter is then used to generate the quarter-pel locations. The data path of the luma interpolator is made up of 6x9 8-bit registers, (4+9) 6-tap filters, and four bilinear filters. The interpolator uses a 6-stage pipeline. At the input of the pipeline, for vertical interpolation, a column of 9 pixels is read from the frame buffer and used to interpolate a column of 4 pixels. A total of 9 pixels, representing the full and half-pel locations are stored at every stage of the interpolator pipeline; specifically, the 4 interpolated half-pel pixels and the 5 center (positions 2 to 6) full-pel pixels of the 9 pixels from the frame buffer are stored. The 9 registers from the 6 stages are fed to 9 horizontal interpolators. Finally, 9:2 muxes are used to select two pixels located
at full or half-pixel locations as inputs to the bilinear interpolator for quarter-pel resolution.

### 2.2.2 Parallelism Analysis

The 4x4 block based row by row interpolation requires at most 288 clock cycles for processing one MB, which cannot meet the requirement of 4Kx2K application. In order to increase the throughput, one solution is to expand the row of 4 pixels to 8 pixels (horizontal expansion), as shown in Figure 2-2.

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 cannot be improved. Moreover, when expanding one row from 8 pixels to 16 pixels, this method results in almost no improvement on the throughput.

![Diagram of horizontal expansion](image-url)
Another way to increase the throughput is to process two or more rows in parallel (vertical expansion), as shown in Figure 2-3. 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 parallelized, the data alignment problem becomes more serious.

![Figure 2-3   Vertical expansion](image)

For example, as shown in Figure 2-4, 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. [16]. However, the corresponding internal memory organization and data control can be very complicated.
2.2.3 Horizontal-Vertical Expansion and Luma-Chroma Parallelism

In order 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. One limit is that for level 3 or higher, which is designed for video specifications higher than or equal to 720x576@25fps, biprediction 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 2.3.3, can be utilized for the horizontal expansion method to reduce the memory data width. However, 8-pixel-parallel processing still cannot 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 2-5. 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.

In order to further enhance the throughput, the interpolation of luma and chroma samples is 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.

2.3 Cache Architecture

The 4Kx2K MC Cache design is mainly challenged by the following aspects. Firstly, with higher specifications, memory bandwidth requirement increases significantly. Previous contributions proved that the cache systems [10][11][12][13][14] can be an effective way to reduce the external DRAM bandwidth. 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.

Furthermore, 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 since higher-speed DRAM specifications such as DDR2 and DDR3. The CAS latency in terms of absolute time is almost the same among DRAM generations, but with the clock frequency increased, the CAS latency in terms of number of clock cycles increases, which influences the cache architecture. On the other hand, new techniques adopted to enhance the DRAM access efficiency, such as reference frame recompression employed in [15], 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, in order 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 2.3.4. 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.

In order to solve the above problems, two schemes are proposed in this section. Firstly, an efficient cache memory organization scheme (4Sx4) is adopted to improve the on-chip memory utilization. By applying this scheme, memory power is saved by 39%~49%, and memory area is reduced by 25%. Secondly, by employing a Split Task Queue (STQ) architecture, the cache system is 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%.

2.3.1 Cache System Design

As shown in Figure 2-6, 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 2.2.2 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 [15].

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 section 2.3.2, and the proposed internal memory organization is presented in section 2.3.3.

2.3.2 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,
A cache based architecture is utilized for reusing the overlapped reference samples of neighboring blocks.

Figure 2-7 shows the 2-set 2x2-MBs sized 2-D cache for this work, which is similar to the design in [14]. 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 [15], 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 [14]. The other difference from [14] 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 2-7 (b).

Moreover, Partial-MB reordering (PMBR) applied in our whole decoder [15] 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 [17] scheme.

2.3.3 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.

![8Sx2 internal memory organization](image)
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. [14] propose an interlaced storage format to buffer the AUs in two 64-bit (8-sample) wide RAMs (hereafter as 8Sx2). As shown in Figure 2-8, 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 2.2.2, there are two ways to increase the interpolation throughput. One is vertical expansion, as shown in Figure 2-9. 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 2-10, when the horizontal neighboring two 4x4 blocks have the same MV (or in one partition), at most 13 pixels for each row are required 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.

![Diagram showing RAM organization with horizontal expansion](image)

*Figure 2-10  4Sx4 internal memory organization with horizontal expansion*

Based on this 4Sx4 scheme and the interpolator described in section 2.2.3 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 2-7 (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.
2.3.4 Split Task Queue Architecture

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

Figure 2-11 shows the previous cache architecture proposed in [14]. 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, in order 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, 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 2-12. 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.

![Diagram of Cache Architecture with STQ](image)

Figure 2-12  Cache architecture with STQ

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.
2.4 System Architecture

The whole motion compensation architecture is illustrated in Figure 2-13. Firstly, tasks which describe the location and size of reference block are sent to the JUDGE unit which judges miss or hit of AUs inside the reference block according to the TAG RAM.

![Motion compensation system architecture diagram](image)

**Figure 2-13**  Motion compensation system architecture

Then, JUDGE unit sends task information to the DUT queue and the addresses of missed AUs to the request queue (REQ queue). According to these addresses, DRAM read requests are generated and burst length of requests is checked which is utilized...
for vertically successive AUs. Together with the DRAM controller employed in our decoder which was described in [18], burst command can reduce the DRAM latency.

When the data is fetched back from DRAM, the RECEIVE unit should detect whether all the missed AUs in the current task are available. So, a dirty code stored in DUT queue is applied for AUs ready detection. For each task, the dirty code represents the situation of required AUs. If the AU is missed, the corresponding bit in dirty code is noted as 0, and otherwise, 1 is noted. Therefore, when the missed AU is received, the corresponding position in the dirty code is noted as 1, and until the dirty code is all 1 which means the required AUs are available, the task is finished.

After conflict checking, the finished task is sent to DRT queue. Then, when the interpolation unit is ready, the task in DRT queue is sent out and the required data for interpolation is output.

Based on the proposed internal memory ordination, the interpolation unit can receive required data for 8x8/8x4 based two row parallel processing in one cycle.

### 2.5 Extended from H.264/AVC to HEVC

There are three proposals in motion compensation part. Firstly, the HEV-LCP parallelism which is proposed to efficiently increase the processing throughput can be directly applied for HEVC.

Secondly, with small modifications, the proposed 8x8-block based 4Sx4 cache memory organization scheme still can be adopted for HEVC. In HEVC, the interpolation filter is change from 6-tap to 8-tap, and 10Sx2 is required for 4x4-block based architectures, as shown in Figure 2-14. The proposed 8x8-block based 4Sx4 scheme can be modified to 5Sx4 to meet the requirement of HEVC. Compared with four 10Sx2 (4x10x2), two 5Sx4 (2x5x4) still can save the memory power in HEVC.
Finally, STQ architecture can be directly applied for HEVC.

![Diagram of 4-pixel based interpolation in HEVC](image)

![Diagram of 8-pixel based interpolation in HEVC](image)

### 2.6 Implementation Results and Comparison

The proposed architecture is implemented in Verilog HDL on RTL level, and then synthesized with Synopsys Design-Compiler 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 [15].
2.6.1 Interpolation Performance

In order to efficiently enhance the interpolation throughput, an HVE-LCP scheme is proposed.

Table 2-1 Interpolation throughput

<table>
<thead>
<tr>
<th>Sequences 1)</th>
<th>QP</th>
<th>Inter MB numbers</th>
<th>Average speed 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IntoTrees</td>
<td>24</td>
<td>249236</td>
<td>44.87</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>211555</td>
<td>36.72</td>
</tr>
<tr>
<td>CrowdRun</td>
<td>24</td>
<td>218336</td>
<td>39.66</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>254880</td>
<td>32.21</td>
</tr>
<tr>
<td>Ducks</td>
<td>24</td>
<td>214360</td>
<td>44.87</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>222770</td>
<td>33.75</td>
</tr>
<tr>
<td>ParkJoy</td>
<td>24</td>
<td>156061</td>
<td>35.93</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>177334</td>
<td>31.17</td>
</tr>
</tbody>
</table>

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

Table 2-1 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 biprediction 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, on 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.

<table>
<thead>
<tr>
<th>Partition</th>
<th>Numbers</th>
<th>Worst-case</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x16 Bi</td>
<td>1</td>
<td>56 cycles</td>
</tr>
<tr>
<td>4x4</td>
<td>12</td>
<td>5 cycles</td>
</tr>
<tr>
<td>8x8 Bi</td>
<td>1</td>
<td>14 cycles</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>130 cycles</td>
</tr>
</tbody>
</table>

Figure 2-16  Example of worst case per two consecutive MBs

Figure 2-17  Interpolation throughput analysis.

Figure 2-16 shows an example of worst case for two consecutive MBs. Hence, the worst case on average processing time for each MB is 65 cycles. Figure 2-17 shows the detail result of the IntoTrees sequence with QP24 and IPBB structure which has the worst performance in Table 2-1. As shown in this figure, most of the MBs require 56 cycles, and the average processing is 48 cycles/MB. The speed requirement in our whole pipelined 4Kx2K decoder is 64 cycles/MB, which is described in [15].
The average speed of the proposed interpolation, and even 58 cycles/MB which has the highest occurrence probability, can meet the requirement.

2.6.2 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 um². 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 153836 um², which is much larger than ours.

Table 2-2 shows the power comparison between the proposed 4Sx4 based memory organization and 8Sx2 based one. For our work, the reading power reduction is 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 2-2 Memory power comparison

<table>
<thead>
<tr>
<th>Seq.</th>
<th>QP</th>
<th>8Sx2 scheme [14]<strong>2</strong> (mW)</th>
<th>Proposed 4Sx4<strong>3</strong> (mW)</th>
<th>Power Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reading</td>
<td>Writing</td>
<td>Total</td>
</tr>
<tr>
<td>IntoTree</td>
<td>24</td>
<td>19.56</td>
<td>2.27</td>
<td>21.83</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>15.51</td>
<td>0.72</td>
<td>16.24</td>
</tr>
<tr>
<td>CrowdRun</td>
<td>24</td>
<td>15.72</td>
<td>0.80</td>
<td>16.52</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>15.54</td>
<td>0.75</td>
<td>16.29</td>
</tr>
<tr>
<td>Ducks</td>
<td>24</td>
<td>16.03</td>
<td>1.10</td>
<td>17.12</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>15.00</td>
<td>0.76</td>
<td>15.76</td>
</tr>
<tr>
<td>ParkJoy</td>
<td>24</td>
<td>10.23</td>
<td>0.54</td>
<td>10.76</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>18.84</td>
<td>0.52</td>
<td>11.35</td>
</tr>
</tbody>
</table>

1): All the sequences are 3840x2160, 10 frames, 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

#### 2.6.3 Overall Performance

STQ Scheme is proposed to reduce the influence from longer memory system latency, and then decrease the area and cache idle time.

In the previous cache structure for comparison, as shown in Figure 2-11, 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 2-12, 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%.

Besides the low area cost, the STQ architecture can significantly reduce the idle time, which contributes to reducing the overall processing time. Table 2-3 shows that the decoding time reduction is from 24% to 40%, compared with the architecture without STQ.

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Without STQ (ms)</th>
<th>With STQ (ms)</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IntoTree</td>
<td>258.26</td>
<td>158.46</td>
<td>-38.64%</td>
</tr>
<tr>
<td>CrowdRun</td>
<td>260.44</td>
<td>156.96</td>
<td>-39.73%</td>
</tr>
<tr>
<td>Ducks²)</td>
<td>257.20</td>
<td>174.50</td>
<td>-32.15%</td>
</tr>
<tr>
<td>ParkJoy</td>
<td>190.69</td>
<td>145.85</td>
<td>-23.51%</td>
</tr>
</tbody>
</table>

1): All the sequences are 3840x2160, 10 frames, IBBP, QP24, running at 166MHz
2): The decoder system cannot support real-time decoding of this sequence because when the QP is 24, the bit-rate of this sequence is larger than the specification of entropy decoding. So entropy decoding component becomes the bottleneck of the whole decoder instead of MC. When the QP is larger, the whole decoder can support real-time decoding of this sequence.

The details of every frame on the IntoTrees sequence with QP24, and IPBB structure running at 166MHz are shown in Figure 2-18. For this sequence, compared with the architecture without STQ, the cache idle time is reduced by about 90%. In a whole system, the processing time for P frame is decreased by about 22%, and B frame decoding time reduction is about 46%. Consequently, the average processing time is saved by 39%.
High-Throughput Motion Compensation

(a) Idle time comparison

(b) Decoding time comparison

Figure 2-18  Cache efficiency comparison
**Table 2.4: Comparison between this work and state-of-the-art MC architectures**

<table>
<thead>
<tr>
<th>Interpolation throughput (Worst-case cycles/MB)</th>
<th>Memory size</th>
<th>Cache gate count</th>
<th>Interpolation gate count</th>
<th>Cache gate count</th>
<th>Technology</th>
<th>Max Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>288 (384)</td>
<td>TP 3.1kB</td>
<td>650</td>
<td>N/A</td>
<td>31k</td>
<td>130nm</td>
<td>1920x1080@30fps</td>
</tr>
<tr>
<td>288 (384)</td>
<td>SP 1.5kB</td>
<td>600</td>
<td>N/A</td>
<td>9k</td>
<td>180nm</td>
<td>1920x1080@30fps</td>
</tr>
<tr>
<td>288 (384)</td>
<td>TP 4kB</td>
<td>560</td>
<td>N/A</td>
<td>72k</td>
<td>90nm</td>
<td>4096x2160@24fps</td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>288 (384)</td>
<td>TP 4kB</td>
<td>72k</td>
<td>N/A</td>
<td>1920x1080@30fps</td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>600</td>
<td>N/A</td>
<td>72k</td>
<td>N/A</td>
<td>1920x1080@30fps</td>
</tr>
<tr>
<td>600</td>
<td>N/A</td>
<td>600</td>
<td>N/A</td>
<td>72k</td>
<td>N/A</td>
<td>1920x1080@30fps</td>
</tr>
<tr>
<td>600</td>
<td>N/A</td>
<td>600</td>
<td>N/A</td>
<td>72k</td>
<td>N/A</td>
<td>1920x1080@30fps</td>
</tr>
<tr>
<td>600</td>
<td>N/A</td>
<td>600</td>
<td>N/A</td>
<td>72k</td>
<td>N/A</td>
<td>1920x1080@30fps</td>
</tr>
</tbody>
</table>

1. The cache gate count and max specification are from [20] and [13], respectively.
2. The cache gate count is composed of 11h for cache and 4h 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 maximum MV number and bi-prediction limits on high levels, the throughput is 288 cycles/MB, if not, it is 384 cycles/MB.
5. Considering the bi-prediction limits on high level, the throughput is 288 cycles/MB.
6. Considering the maximum MV number and bi-prediction limits on high levels, and worst case on per two consecutive MBs is 130.
2.6.4 Whole Architecture Performance Comparison

A comparison between this architecture and state-of-the-art works is shown in Table 2-4. 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 130nm 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 powers are optimized. Finally, with the DQT scheme, our design can tolerant longer memory system latency and reduce the decoding time of whole system.

2.7 Summary

In this chapter, three schemes are proposed to achieve an efficient MC architecture for H.264/AVC real-time decoding of ultra high definition application.

In section 2.1, the main design challenges are presented including the huge area cost and power consumption. Moreover, the long memory system latency leads to performance drop of the MC pipeline.

In section 2.2, 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.

In section 2.3, an efficient cache memory organization scheme (4Sx4) is adopted to improve the on-chip memory utilization, which contributes to memory area saving
of 25% and memory power saving of 39%~49%. Moreover, by employing STQ architecture, the cache system is capable of tolerating much longer latency of the memory system.

In section 2.4, the overall MC architecture is summarized. The techniques about using request queue and dirty code are illustrated.

Section 2.6 shows the experimental results. 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 4kx2k video decoder architecture [15].
3 Efficient Joint Parameter Decoder

Motion data decoding, which derives the current motion vector and reference index, is one of the most algorithm-irregular components of the inter-frame decoding in H.264/AVC framework due to the use of variable-block-size inter prediction and multiple prediction modes. Hence, efficient hardware design of this component can be very difficult, especially for ultra-high resolution applications.

In section 3.1, we propose to combine motion data calculation, together with intra prediction mode calculation, bound strength calculation as a joint parameter decoder.

In section 3.2, we analyze the problems in pipeline design and discuss the problem solutions in state-of-the-art works. Then, a more hardware efficient implementation by taking advantage of the high-level features of H.264/AVC standard is proposed.

In section 3.3, three steps DRAM bandwidth reduction strategy is presented. In this strategy, a partition based storage format is first applied to condense the MB level data. Then, a DPCM based variable length coding method is utilized to reduce the data size of each partition. Finally, the total bandwidth is further reduced by combining the co-located and last-line information.

In section 3.4, the system architecture which is composed of pipelined calculation part and data compression/decompression part is described. The connection method between the two parts is presented.
Section 3.6 shows the implementation result and comparison with state-of-the-art works. It shows the comparison results on throughput, DRAM bandwidth, hardware cost, and functions, respectively.

3.1 Design Consideration

In our work, motion data (MD) calculation, together with the intra prediction mode (IPM) calculation and boundary strength (BS) calculation are implemented as a joint parameter decoder (JPDEC) architecture, based on the following considerations.

On one hand, by combining the IPM calculation and MD calculation components, the function of reading and writing the neighboring data can be reused since both of them require data from the adjacent MBs. On the other hand, integrating the calculation of BS and MD helps eliminate the buffer for transmitting the current MDs, and also enables the sharing of neighboring MD fetching operation, because these MDs are required for BS calculation.

3.1.1 Processing Cycles

The design of JPDEC architecture is mainly challenged by the application of variable-block-size and multiple-mode inter/intra prediction, which means the computation complexity varies significantly for MBs with different coding features. MD calculation is one of the most algorithm-irregular components of the H.264/AVC framework. Hence, efficient hardware design of this component can be very difficult, especially for ultra-high resolution applications. It is also the most complex part of our JPDEC architecture.

Most of the previous works on MD calculation architectures have been targeted to standard HD applications. In Yoo et al.’s work [21], the processing time of the
motion vector processor vary from 28 cycles/MB to 260 cycles/MB. Another architecture proposed by Yin et al. [22], takes 260 cycles to process one MB for the worst-case, and 160 cycles for average.

Considering these architectures' overall speed in a pipelined video decoder is restricted by the worst-case performance, it is difficult to apply them to higher specifications. In order to improve the worst-case performance of the MD calculation part, this paper presents designs for reducing the processing time and control complexity. A joint pipeline mechanism is designed to save the processing time. And then, through categorizing the various partition sizes and prediction modes into two groups, the control logic is reduced and the pipeline stall on partition size alteration can be avoided. As a result, a high constant throughput pipeline mechanism is achieved.

3.1.2 External Memory Traffic

With specifications enhanced, data bandwidth will also bottleneck the design of video codec, for the video system usually rely on large external DRAMs for buffering the mass data such as reference frames and co-located information. Owing to the huge bandwidth requirement, power consumption for DRAM traffic can compose a significant portion of the system power, especially for ultra-high definition applications.

Many works have been conducted to reduce the bandwidth requirement of motion compensation (MC) part [23][24][25][26][27][7], which originally composes a dominant bandwidth portion of the whole decoder system. Chen [23] and Chuang[24], etc. developed efficient cache architectures for reusing reference data, which contributed to 70%~80% reduction of MC read access. [28][29][30][31][32][33] presented various methods for compressing the reference
frames, so that 40%~60% bandwidth for both MC/ME read and write access can be saved.

<table>
<thead>
<tr>
<th>Frame write (Mbytes)</th>
<th>Frame read (Mbytes)</th>
<th>Col write/read (Mbytes)</th>
<th>Total (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>60.2 (34.4%)</td>
<td>79.4 (45.4%)</td>
<td>35.3 (20.2%)</td>
<td>174.9 (100%)</td>
</tr>
</tbody>
</table>

Note: The result is on 10-frame. 3840x2160 IntoTrees.264, IBBP, QP32.

Meanwhile, few works have been found on the bandwidth optimization for the other parts. However, considering that the MC portion has already been reduced by almost 90%, the non-MC parts of the total bandwidth, especially the co-located motion information (motion vector and reference index), can no longer be taken as negligible. Table 3-1 shows the bandwidth profiling result for decoding a 2160p sequence when the bandwidth reduction in frame read and write are considered. By taking into consideration the latest optimization techniques, the bandwidth for accessing reference frames drops to be less than 4 times as that for the non-optimized co-located information. As a result, co-located information access can now compose over 20% of the total DRAM bandwidth, if the rest (other than reference frames and co-located motion) memory access is performed on on-chip SRAM. However, the existing work on the motion vector calculation [21][22], did not consider reducing the co-located information.

The generally believed requirement for “random accessibility” is one of the reasons that make bandwidth optimization for co-located information difficult: since only the B Skip MBs require data from their co-located positions, it is natural to think that the data at “useless” positions should never be fetched from DRAM so as to make efficient use of the bandwidth resource. A simple and commonly used solution to this requirement is to store the MV and refIdx for each 4x4-block independently with
fixed data length, although it meanwhile results in massive data redundancy. Contrarily, in this paper, we propose to store the co-located information in a partition-by-partition format. Obviously, this leads to the sacrifice of random accessibility, which means for decoding each B frame, all its co-located information in DRAM should be traversed, regardless of which part is really “useful”. But actually, according to experimental results, the use of efficient storage and coding can not only compensate the loss of random accessibility, but also significantly reduce the total bandwidth.

In our work, we propose a DPCM based variable length coding (VLC) to compress the MV and refIdx data. Finally, considering that the last-line motion information, which is also required by MD decoding, shares a considerable amount of identical data with the co-located write bandwidth, a joint method is developed for co-optimizing these two parts. Low-cost VLSI architecture for the proposed optimization methods is implemented as a transparent layer between the motion vector decoder circuits and the DRAM interface. Finally, the DRAM bandwidth for co-located and last-line motion information is successfully reduced by 85%-98%, with an additional area of only 7.8k gates.

3.2 Pipeline Design

The design of JPDEC architecture is mainly challenged by the application of variable-block-size and multiple-mode inter/intra prediction, which means the computation complexity varies significantly for MBs with different coding features. In this section, we discuss the pipeline architecture which improves the throughput by applying a dual-mode pipelining design.
3.2.1 Pipeline Analysis

Taking MD calculation as an example, a non-pipelining implementation (Figure 3-1 (a)) requires at least three stages performed sequentially to derive the MD for a partition, including memory read (Mem Rd.) stage for fetching reference MDs, calculation (Calc. MD) stage for arithmetic operations, and memory write (Mem Wr.) stage for writing back the results. This type of design is easy to implement with a state machine, whereas it suffers from a relatively low throughput because the calculation hardware is not efficiently used.

![Pipeline analysis diagram](image)

(a) Sequential (non-pipelining) solution.

(b) Ideal pipelining solution.

(c) Single-mode pipelining solution

(d) Proposed dual-mode pipelining solution

Figure 3-1 Pipeline architecture analysis

Theoretically, by parallelizing arithmetic and memory operations, the processing flow in Figure 3-1(a) can be scheduled to be as Figure 3-1(b) for eliminating the time for memory read/write. Meanwhile, the classic data forwarding techniques can help bypass the results from one calculation stage to the next, instead of transferring them
through the memory, so as to solve the data dependency problem. However, with various partition sizes and prediction modes applied, each combination of these two factors may require a different cycle count for arithmetic operation, or a different time point to issue the memory operations. Furthermore, the operations need careful scheduling for avoiding resource hazards (e.g. conflicts on use of memory ports) under various conditions, and result forwarding also adds to the control complexity. All these issues result in a very large pipeline controller that may be too difficult and cost ineffective to be implemented, while the situation can be even more complicated if IPM and BS, in addition to MD calculation, are taken into consideration.

To solve these problems, Yoo et al. [21] developed their architecture based on the solution in Figure 3-1(c). For this solution, each partition requires a constant cycle count and fixed sequence for memory and arithmetic operations. Therefore the pipeline control can be significantly simplified. However, this means the simplest and the most complicated prediction modes are allocated with the same processing time, which reduces the hardware efficiency. And the worst-case (when one MB consists of 16 partitions) cycle count for each MB is $16\lambda$, where $\lambda$ is the calculation cycle count for each partition. Although the best-case cycle count is $\lambda$ when there is only one partition in the MB, this hardly helps the design of a decoding system, which should be planned according to the worst-case, or at least, the average performance of its components.

In this work, we try to realize a more hardware efficient implementation by taking advantage of the high-level features of H.264/AVC standard. For level 3 or higher, which is designed for video specifications higher than or equal to 720x576@25fps, bi-prediction MV is not allowed for partition sizes smaller than 8x8, and direct prediction is performed in 8x8, instead of 4x4 partitions. This means the MD calculation of 8x4, 4x8 and 4x4 partitions can be less complicated than that of
the larger ones, and therefore requires less cycle count. Based on this consideration, we proposed a dual-mode pipelining solution as shown in Figure 3-1(d).

For this solution, two types of calculation operations for 4x4 and 8x8 blocks are organized together. While the processing time for a 8x8 block can still be regarded as \( \lambda \) (\( \lambda=16 \) in this implementation), each 4x4 block only needs \( \lambda/4 \). As a result, considering each MB can contain at most 4 8x8 blocks or 16 4x4 blocks, the worst-case cycle count for each MB is reduced to \( 4\lambda \), compared to the \( 16\lambda \) for Figure 3-1(c). By using this dual-mode solution, control logic can be a bit more complicated than the single-mode solution, but still significantly simpler than the ideal pipeline in Figure 3-1(b). Details of the pipeline design, together with the support for IPM and BS calculation, are discussed in section 3.2.2.

### 3.2.2 Dual-mode Pipelining Architecture

Dual-mode pipelining architecture is proposed to reduce the processing time and hardware cost. Through categorizing the various partition sizes and prediction modes into two control modes, the control logic is reduced and the pipeline stall on partition size alteration can be avoided. As a result, a high constant throughput pipeline mechanism is achieved.

Firstly, control mode categorization is proposed. The MD/IPM calculation task, which determines the processing time of each partition, is composed of two operations, MD calculation for inter MB and IPM calculation for intra MB. Between the two operations, the derivation process of MD is more complicated and irregular due to the use of various partition sizes (4x4, 4x8, 8x4, 8x8, 16x8, 8x16, 16x16) and multiple prediction modes (skip, forward, backward, bi-directional, spatial direct and temporal direct). The irregularity results in variable throughput of the whole JPDEC architecture, and consequently may bring overhead to the pipelined video decoder.
Moreover, hardware implementation for such algorithm-irregular component can be very difficult. However, it is difficult to obtain the fixed processing time for each partition, since the partition size and prediction mode, which determine the computational complexity of the partition, is variable.

In order to overcome the above problem, an MB level constant throughput pipeline mechanism is designed. We propose to simplify the processing patterns according to the following considerations on high-level limits described in H.264 standard. Firstly, when the level is higher than 2.2, direct 8x8 inference flag is set equal to 1, which means derivation process of MD on the B Skip, B Direct 16x16, and B Direct 8x8 modes can be processed based on 8x8 block. Secondly, on levels higher than 3, bi-prediction mode is not allowed in partitions smaller than 8x8. Based on the above specifications, the small block sizes, which should be processed more times in one MB, consume less calculation time, because only the single-directional prediction is allowed. Meanwhile, the complex B skip, B direct mode, and bi-prediction are based on large partition sizes.

Therefore, we classify the various partition sizes and prediction modes into two groups, 4x4-block group and 8x8-block group as shown in Table 3-2. 4x4-block based control mode and 8x8-block based control mode is applied to the two groups respectively. For the 4x4-block group, it contains partitions smaller than 8x8 in regular mode. The others including the P skip mode, B Skip, B Direct 16x16, B Direct 8x8, and the partitions with support to bi-prediction in regular mode are classified to 8x8-block group. Although the calculation for 8x8-block based control mode is more complex than the 4x4-block based on, it has four times processing time budget as the 4x4-block based control mode. In order to share the same processing schedule with MD calculation part, the IPM calculation part is classified to the same two control modes as the MD calculation part, for which the 4x4-block based intra processing is
categorized to 4x4-block based control mode while 8x8 and 16x16 are classified to 8x8-block based control mode.

Table 3-2 Control Mode Categorization

<table>
<thead>
<tr>
<th>Control Mode</th>
<th>4x4-block group</th>
<th>8x8-block group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intra Prediction Mode</td>
<td>Intra 4x4</td>
<td>Intra 8x8</td>
</tr>
<tr>
<td>Partition Patterns for inter MB&lt;sup&gt;1)&lt;/sup&gt;</td>
<td>PredL0_4x4, PredL0_8x4, PredL0_4x8, PredL1_4x4, PredL1_8x4, PredL1_4x8</td>
<td>PredL0_8x8, PredL0_8x16, PredL0_16x8, PredL0_16x16, PredL1_8x8, PredL1_8x16, PredL1_16x8, PredL1_16x16, BiPred_8x8, BiPred_8x16, BiPred_16x8, BiPred_16x16, BSkip, BDirect_16x16, BDirect_8x8, P_Skip</td>
</tr>
</tbody>
</table>

<sup>1)</sup>: is represented in this way: PartitionPredMode_PartitionWidth x PartitionHeight.

The pipeline is composed of three tasks including memory read/write, MD/IPM calculation, and BS calculation. The same processing framework is shared to save the logic count, as illustrated in Figure 3-2. All of the three tasks are divided into two control modes, 4x4-block based and 8x8-block based control modes, for being integrated to the designed pipeline schedule. Moreover, processing the Mem Rd. of next block (8x8-based block or 4x4-based block) can be parallelized with MD/IPM Calc of the current block and BS Calc of the previous one, so as to avoid pipeline bubbles at block and MB boundaries.
Based on the control mode categorization, the pipeline schedule is derived. For the 4x4-block based control mode, all the members in this control mode are divided into several 4x4 blocks. For each 4x4 block, the processing time mainly depends on memory reading and writing part since the calculation parts (MD/IPM Calc and BS Calc) are not complicated. In the memory related part, the first line of blocks in one MB require the longest time for fetching back the adjacent information because the information of upper blocks is stored in the external memory, as shown in Figure 3-3. For the other blocks in this MB, the adjacent data can be got instantly, since the neighboring information from either the left MB or the decoded blocks in current MB, is stored in registers. Then, one cycle is needed to write back the information of the previously decoded block. Therefore, the time budget for memory writing and reading is four cycles (three cycles for reading the adjacent data for current block and one cycle for writing back the information of the previous one). Moreover, since the calculation is not complex for the 4x4-block based control mode, the MD calculation for inter MB, IPM calculation for intra one, and BS calculation can all be finished within four cycles.

Figure 3-3  Determination of the neighboring blocks
In the 8x8-block based control mode, the members are segmented into several 8x8 blocks. The similar pipeline structure, which requires sixteen cycles for each 8x8 block, is employed to avoid the stall between the boundaries of different groups.

The pipeline mechanism, as shown in Figure 3-4 brings the following benefits. Firstly, by combing the MD and IPM calculation, the Mem wr. & re. part which prepares the data of neighboring MBs, can be reused for saving the area cost. Secondly, MD calculation and IPM calculation can share the same pipeline to avoid hazards on MB type alteration. Finally, since the three operations of reading/writing the neighboring information for the next block, calculating the MD or IPM for the current one, and calculating BS for the previous one, can be processed at the same time, the processing time of the JPDEC can optimized.

As a result, no matter how an MB is partitioned (sixteen units of 4x4-block group, four units of 8x8-block group, or combination of 4x4-block group and 8x8-block group), and no matter what type (inter or intra) an MB is, a constant-throughput of 64 cycles/MB can be obtained.
Efficient Joint Parameter Decoder

Figure 3.4
JPDEC pipeline mechanism for different control modes

Note: Deeper rectangles stand for the data of current block, with lighter ones for previous or next block. 4 cycles and 16 cycles are required to process every 4x4 and 8x8 block, respectively. The constant throughput of every MB is 64 cycles.

rd: read memory, wr: write back to mem, res: get the result of current block.

(a) 8x8-block based control mode

(b) 4x4-block based control mode
3.3 DRAM Bandwidth Reduction Strategy

In order to reduce the DRAM bandwidth, we propose three steps bandwidth reduction strategy. On step 1, a partition based storage format is applied to condense the MB level data. On step 2, variable length coding based compression method is utilized to reduce the data size in each partition. Finally, the total bandwidth is further reduced by combining the co-located and last-line information. The following sections describe the three steps in detail.

3.3.1 Partition Based Storage Format

Generally, in order to support random access of the co-located information for avoiding extra memory traffic when only a certain part of data is required, the co-located information is stored for each 4x4 block with fixed data length. For this method, even when the size of partition, which is the basic processing unit of inter prediction, is larger than 4x4, the same co-located information is repeatedly stored. Consequently, the large memory size leads to large DRAM bandwidth.

In this work, one partition is utilized as the basic storage unit to eliminate the data overlapping and reduce the memory writing bandwidth. However, the variable partition size brings data dependency between MBs. Hence, random access can no longer be supported, so that the stored data for a whole co-located picture should be fetched back even when only a small part of it is required. Although the extra memory reading traffic is incurred, the long latency of DRAM access can be avoided. Moreover, especially for high specification applications which require very large memory bandwidth, the partition based storage format is more efficient because of the following reasons.
On one hand, as designed in the H.264 standard on levels higher than 3.0, the maximum number of motion vectors per two consecutive MBs should be less than 16. Accordingly, the average number of partitions for each MB is smaller than 8, which consequently results in more than half memory writing reduction, compared to the original 4x4-based storage format.

On the other hand, B skip and B direct modes which requires the co-located information, occupies high proportion in usual, so the reading bandwidth increase cannot be significant.

Figure 3-5 shows the detailed storage format. When the partition size is larger than 8x8, 2 bits are used to represent the partition size for each MB. Otherwise, 10 bits are needed. Then, the co-located information of each partition is stored in order. Since the bit cost of the co-located information for one MB is from 29 to 464 bits (29 bits for 16x16 mb_type, and the maximum partition number for one MB is 16), the bits denoting partition sizes (3%~7% of the total size) are likely to be negligible. Another extra one bit which denotes the intra MB is required for the last-line information. When the current MB is in intra mode, this intra MB flag is set to one, the partition size is set to intra prediction size, and the subsequent information is replaced by intra mode.
Efficient Joint Parameter Decoder

Sample Microblock: There are many different ways for partitioning one MB. This is just an example.

ITF*: Intra flag. This flag is only used for last line information compression. When the current MB is intra, this flag is set to one. And the following bits are used to represent the intra mode instead of the MV and RefIdx.

Step 1: Partition-based storage and DPCM-based variable length coding

```
Par. 0
4x4 block 0
Original Storage
4x4 block 1
4x4 block 2
4x4 block 3
4x4 block 4
4x4 block 14
4x4 block 15
```

Step 2: Exp-Golomb Coding

```
 Unary Coding
```

Step 3: Unary Coding

```
Par. 1
```

 Steph 4: Partition-based storage

```
```

Step 5: Partition-based storage

```
```

Figure 3.5
3.3.2 DPCM Based Variable Length Coding

Both the co-located and last-line motion information is composed of two parts including motion vector and reference indices. According to the different properties of the two parts, two variable length coding (VLC) methods are applied to encode them respectively.

For reducing the spatial redundancy, the MVs are first DPCM-coded, by using the difference between the current and the last calculated MVs. Considering that the dependency is not obvious between MVs with different reference indices, DPCM coding is performed only when the current and the last MVs share the same reference index.

As shown in Figure 3-6, the probability distribution of MV DPCM value is usually similar to geometric distribution. Hence, the Exp-Golomb coding (Table 3-3) is proceeded to these values so as to express the co-located MVs with less bits. If refIdx is different between the current and the previous partitions, the DPCM value becomes large which deteriorates the efficiency of Exp-Golomb coding. Therefore, for this case, the original MV values, instead of MV DPCM values, are stored. When the co-located information is fetched, the MVs are restored in the identical sequence in which they have been stored.
Figure 3-6  Probability Distribution of MV DPCM and Exp-Golomb Code

Table 3-3  Signed Exp-Colomb code

<table>
<thead>
<tr>
<th>N</th>
<th>CodeNum</th>
<th>Exp-Golomb</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>010</td>
<td>0.125</td>
</tr>
<tr>
<td>-1</td>
<td>2</td>
<td>011</td>
<td>0.125</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>00100</td>
<td>0.03125</td>
</tr>
<tr>
<td>-2</td>
<td>4</td>
<td>00101</td>
<td>0.03125</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>00110</td>
<td>0.03125</td>
</tr>
<tr>
<td>-3</td>
<td>6</td>
<td>00111</td>
<td>0.03125</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

As a result, the restoring component is capable of identifying whether MV DPCM value or original MV has been stored, based on whether or not the refIdx of the current partition equals to that of the last restored partition. This means no extra flags are required to indicate which one is used.

Figure 3-7 shows the probability distribution of reference indices, among which most of the values are very close to zero. Hence, unary coding (Table 3-4) is adopted for them.
3.3.3 Integrated Coding for Co-located and Last-line Data

The partition based storage format and VLC method which are applied to compress the data of co-located information, can also be utilized for the last-line information, as shown in Figure 3-8.

Although the potential for usable dependency between the partitions in the last-line information buffer is lower, there is no extra memory traffic for not supporting random access since all the data in the last-line buffer should be fetched back for subsequent decoding. As a result, the memory traffic for processing the last-line buffer can also be significantly reduced.
Sample Microblock*: there are many different ways for partitioning one MB, this is just an example.

ITF*: Intra flag. When the current MB is intra, this flag is set to one, and the following bits are used to represent the intra mode instead of the MV and RefIdx.

Moreover, considering the last-line information is actually a subset of the co-located information in P frames, some data are stored two times (both in the co-located last-line buffers). The straightforward way to eliminate the data overlapping is removing the last-line buffer for P frame and fetching the last-line information directly from the co-located buffer. However, with this method, all the co-located information should be fetched back for use as last-line data, due to the data dependency between the partitions, while only a part of it is really required. A lot of useless data fetching may result in increasing, instead of reducing the memory traffic.

Therefore, the following organization is designed to reuse the overlapped data, and avoid extra memory access overhead. Firstly, as shown in Figure 3-8, the co-located and last-line buffers are combined together to store the data of the whole line.
co-located picture. If the data of one partition is stored in last-line buffer, it will not be stored in the co-located information buffer again. Hence, every time for fetching back the co-located information, the last-line buffer is read first, and then, the co-located buffer is accessed for getting additional data, except for the mb type is 16x16 or 8x16 for which all the data are already buffered in the last-line buffer. Furthermore, to prevent from affecting the compression ratio while lowering the data dependency potential, the partitions in co-located buffer can refer to the previous partitions even if it is in the last-line buffer. Meanwhile, the partitions in the last-line buffer can only refer to the previous ones in the same buffer, and as a result, the data dependency potential within the line information is slightly lowered. However, on the sacrifice of the compression ratio, the extra memory traffic caused by mass useless data being fetched back can be removed.

3.4 System Architecture

The whole JPDEC system is composed of two parts, calculation part and compression/decompression part. A dual-mode pipeline is applied in the calculation part to improve the throughput, and three step DRAM bandwidth reduction strategy is employed in the compression/decompression part.

The connection method of these two parts is shown in Figure 3-9. Two FIFOs connect the pipelined component and compression/decompression part. For writing data from FIFO to the DRAM, the last-line and co-located information of current frame are compressed by joint coding, and then, the compressed data is stored into DRAM. For the required data of MD/IPM calculation, the last-line information of current frame and co-located frame, together with the co-located information of co-located frame, are fetched back from DRAM, after that, the decompressed data is
sent to FIFO which is connected to the calculation part. Since the data compression/decompression behaviors and pipelined calculation part are separated by FIFOs, the processing time and critical path of the whole JPDEC system will not be influenced from the compression scheme.

![Diagram showing data flow between DRAM and pipelined calculation part](image)

**Figure 3-9** Connection between DRAM and pipelined calculation part

### 3.5 Extended from H.264/AVC to HEVC

There are also three proposals in joint parameter part. Firstly, the idea of combining three functions to save memory size and area cost still can be applied for HEVC. Secondly, since more partition sizes and prediction modes are added in HEVC, the proposed mode classification should be changed. With the modified mode classification, the proposed dual-mode pipeline can be utilized for HEVC. Finally, the bandwidth reduction strategy can be directly adopted for HEVC.
3.6 Implementation Results and Comparison

The proposed architecture is implemented in Verilog HDL on RTL level, and then synthesized with Synopsys Design Compiler by using SMIC 90 G standard cell library. Under a timing constraint of 200MHz, synthesis result shows a logic gate count of 37.2k. This design is verified both independently in a test environment with inputs given as software generated data, and in a whole 2160p video decoder architecture.

3.6.1 Throughput Comparison

A comparison between this architecture and state-of-the-art works is shown in Table 3-5. Compared with the stat-of-art architectures, by applying the dual-mode pipelining architecture, there are two advantages in our work. Firstly, the number of clock cycles required for processing one MB is reduced by 75% at the worst case. Moreover, through categorizing the various partition sizes and prediction modes into two groups according to the high-level limits specified in H.264 standard, MB level constant throughput architecture is obtained.

Table 3-5 Comparison between this work and state-of-the-art MD architectures

<table>
<thead>
<tr>
<th></th>
<th>[21]</th>
<th>[22]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst-case, Cycles/MB</td>
<td>260</td>
<td>260</td>
<td>64</td>
</tr>
<tr>
<td>Avg. Cycles/MB</td>
<td>N/A</td>
<td>160</td>
<td>64</td>
</tr>
<tr>
<td>Operating Frequency</td>
<td>266MHz</td>
<td>N/A</td>
<td>200MHz</td>
</tr>
<tr>
<td>Supported Function</td>
<td>MD calculation</td>
<td>MD calculation</td>
<td>MD calculation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IPM calculation</td>
</tr>
<tr>
<td>Gate Count</td>
<td>41.1k</td>
<td>52k</td>
<td>37.2k*</td>
</tr>
<tr>
<td>Technology</td>
<td>65nm</td>
<td>0.18um</td>
<td>90nm</td>
</tr>
</tbody>
</table>

37.2k*: Consisting of 3.1k for compressing the DRAM bandwidth, 4.7 for decompressing, and 29.1k for the calculation part.
3.6.2 DRAM Bandwidth Comparison

Table 3-6 shows the bandwidth test result on various sequences. Firstly, the partition based storage format is applied. Compared with the general 4x4-block based storage, the proposed storage format leads to the sacrifice of random accessibility, which means for decoding each B frame, all its co-located information in DRAM should be traversed, regardless of which part is really “useful”. But actually, according to experimental results which take into account this overhead, the partition based storage can not only compensate the loss of random accessibility, but also significantly reduce the total bandwidth, especially for high-level sequences. Then, the DPCM-based variable length coding including the Exp-Golomb coding and unary coding, is utilized to further condense the data stored by partition. By combing the compressed co-located and line information buffer, the DRAM bandwidth can be finally reduced by 85%-98%. As described in section 3.1, this can contribute to a maximum of 20% bandwidth reduction for the whole video decoder system.
Table 3-6 Bandwidth test results on various sequences

<table>
<thead>
<tr>
<th>Sequence</th>
<th>(3840x2640)</th>
<th>(10frms)</th>
<th>(3840x2640)</th>
<th>(10frms)</th>
<th>(3840x2640)</th>
<th>(10frms)</th>
<th>(1080p)</th>
<th>(10frms)</th>
<th>(1080p)</th>
<th>(10frms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParkJoy</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
</tr>
<tr>
<td>CrowRun</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
</tr>
<tr>
<td>InToTree</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
<td>24</td>
<td>IBBP</td>
</tr>
<tr>
<td>Station</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
</tr>
<tr>
<td>BlueSky</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
<td>24</td>
<td>IPPP</td>
</tr>
</tbody>
</table>

QP

<table>
<thead>
<tr>
<th>Structure</th>
<th>GOP</th>
<th>Original (M Bits)</th>
<th>PBS 2)</th>
<th>VLC 3)</th>
<th>INC 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>194.65</td>
<td>IBBP</td>
<td>55.32</td>
<td>4.57</td>
<td>1.86</td>
<td>3.80</td>
</tr>
<tr>
<td>101.21</td>
<td>1080p</td>
<td>55.32</td>
<td>4.57</td>
<td>1.86</td>
<td>3.80</td>
</tr>
<tr>
<td>32.40</td>
<td>IPPP</td>
<td>119.15</td>
<td>24.67</td>
<td>3.18</td>
<td>1.31</td>
</tr>
<tr>
<td>30.19</td>
<td>IPPP</td>
<td>264.21</td>
<td>26.41</td>
<td>3.18</td>
<td>1.31</td>
</tr>
<tr>
<td>8.52</td>
<td>IPPP</td>
<td>119.15</td>
<td>24.67</td>
<td>3.18</td>
<td>1.31</td>
</tr>
<tr>
<td>8.52</td>
<td>IPPP</td>
<td>264.21</td>
<td>26.41</td>
<td>3.18</td>
<td>1.31</td>
</tr>
</tbody>
</table>

△BW

1) The original uncompressed data stored by 4x4 blocks. 2) Step 1: partition based storage. 3) Step 2: DPCM-based variable length coding. 4) Step 3: integrated coding for co-located and last-line data. 5) The overhead of losing random access flexibility is taken into account for the compression method.
3.6.3 Hardware Cost Comparison

As shown in Table 3-5, the total gate count of proposed architecture is also the smallest one due to the simplified control modes.

The total gate count of proposed architecture is also the smallest one due to the simplified control modes. The detail gate count distribution is summarized in Table 3-7. The area cost for compression and decompression which reduces 85%-98% memory bandwidth, occupies only 20% of the total gate count due to the selected Exp-Golomb code and unary code are suitable and simple.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Gate Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem wr. &amp; re.</td>
<td>3.8k</td>
</tr>
<tr>
<td>MD/IPM calculation</td>
<td>18.3k</td>
</tr>
<tr>
<td>BS calculation</td>
<td>1.8k</td>
</tr>
<tr>
<td>Register Control</td>
<td>4.6k</td>
</tr>
<tr>
<td>Compression</td>
<td>3.1k</td>
</tr>
<tr>
<td>Decompression</td>
<td>4.7k</td>
</tr>
<tr>
<td>Total</td>
<td>37.2k</td>
</tr>
</tbody>
</table>

3.6.4 Function Comparison

As shown in Table 3-5, another merit of this architecture is that the MV calculation, intra prediction mode derivation and BS computation are combined into a single architecture, for which the operation of fetching the neighboring information can be shared between and MV and IPM calculation components, and the MV buffer between the MV and BS calculation components is eliminated.
3.7 Summary

In this chapter, VLSI architecture of a joint parameter decoder is proposed. For this architecture, a 64-cycle-per-MB pipeline with simplified control modes is designed to increase system throughput and reduce hardware cost. Moreover, in order to save memory bandwidth, three-step memory bandwidth reduction strategy is applied.

In section 3.1, processing speed and DRAM traffic problem are presented. And then, JPDEC system is proposed to combine MD, IPM and BS for H.264/AVC standard.

In section 3.2, to achieve an efficient design, dual-mode pipelining architecture is applied. Through categorizing the various partition sizes and prediction modes into two control modes, the control logic is reduced and the pipeline stall on partition size alteration can be avoided. As a result, a 64-cycle-per-MB pipeline is designed to integrate the three functions, so that 2160p@60fps sequences can be processed with less than 166MHz.

In section 3.3, in order to save the DRAM access, a partition based storage format is first applied to condense the MB level data. Then, a DPCM-based variable length coding method is utilized to reduce the data size of each partition. Finally, the total bandwidth is further reduced by combining the co-located and last-line information. Experimental results show that the bandwidth requirement for motion vector calculation can be reduced by 85%-98% on typical 1080p and 2160p sequences, with only 7.8k additional logic gates. This can contribute to near 20% bandwidth reduction for the whole video decoder system.

In section 3.4, system architecture is described. The pipelined calculation part and data compression/decompression parts are connected by buffers. Hence, the data
compression and decompression behaviors are transparent to the pipelined calculation part, for which the processing time of the whole system will not be restricted from the compression scheme.

Finally, section 3.6 shows the experimental results and compare our work with state-of-the-art architectures.
4 Alternating Asymmetric Search Range Assignment for Motion Estimation

Motion estimation (ME) contributes significantly to video coding efficiency, whereas it is also the most computation intensive component of a video encoder. Most ME architectures in hardware are based on full search or modified versions of full search due to its regular data processing flow. To reduce the complexity of full-search ME which is determined by the search range, we propose alternating asymmetric search range assignment (AASRA) schemes including AASRA-B, AASRA-P and AASRA-PB.

In section 4.1, recent works on motion estimation are introduced including new search patterns and modified full search.

In section 4.2, three schemes including AASRA-B, AASRAP and AASRA-PB are proposed. AASRA-B uses a large and a small search ranges, respectively, for the two reference directions of bidirectional ME. AASRA-P extends the application of AASRA to P frames by alternating the large and small search ranges among consecutive MB/CTBs. AASRA-PB combines the features of AASRA-B and AASRA-P to obtain even higher complexity saving.

In section 4.3, we further propose the necessary adaptations to apply AASRA on the basis of MB-parallel ME and hierarchical ME. This demonstrates the proposed methods can also be combined with existing ME architectures.

In section 4.4, to more appropriately evaluate the effectiveness of the proposed schemes for high-motion sequences, we also propose a method to calculate the equivalent fixed search range (EFSR) for AASRA. According to EFSR, after the
coding performance difference has been fairly removed from comparison, the AASRA-B, AASRA-P and AASRA-PB schemes can still reduce complexity by 34.2%, 28.9% and 43.1%, respectively. Conventional RD-based evaluation of the proposed is also performed.

4.1 Recent Works on Motion Estimation

As profiled in [34], ME takes more than 50% of the total computation time in an H.264/AVC encoder when configured to use single-direction full search and a search range (SR) of 32. This can be even more critical taking the following two factors into account. On one hand, relative to single-direction ME, bidirectional ME is becoming increasingly popular due to its better prediction performance, whereas it also leads to at least doubled complexity. On the other hand, video contents with higher resolutions such as 1080p HD, 4K and 8K Ultra HD (or Super Hi-Vision, SHV) requires a much larger SR to achieve a good compression ratio, while the complexity of full-search ME is approximately proportional to $SR^2$.

Many techniques have been developed to reduce the ME complexity while maintaining coding performance. One category of the techniques is to apply new search patterns instead of full search to reduce the number of search points checked in the search area. Three step search [35], four step search [36], diamond search[37], and cross diamond search [38] are representative among these approaches. They can usually reduce the computation effectively and thereby speed-up software-based encoders. Nevertheless the new search pattern usually involves an irregular data processing flow which is difficult to be pipelined or parallelized in hardware implementation.
In fact, most of the hardware ME architectures, especially those implemented in recently published video encoder chips [39] [40], are based on full search or modified versions of full search. Ding, et al. [40], [41] applied a candidate based search center derivation method to improve the performance of a full-search ME with relatively small search range. Lin, et al.[42], Bao, et al.[43], and Peng, et al. [44]’s hierarchical ME architectures performed full searches on multiple levels of down-sampled references to reduce the complexity for supporting a large search window.

To reduce search range and thereby reduce the complexity of full-search ME, several dynamic search range selection algorithms have been reported [45]-[48]. Their basic idea is to adaptively assign search range according to the expected motion intensity, so that the average computation time can be saved. However, these algorithms cannot ensure a stable complexity saving, and are consequently incapable of improving the worst-case performance which is crucial for real-time systems. Jung, et. al [49]’s dynamic search range adjustment algorithm can achieve a stabilized reduction of memory traffic, but its computational complexity is still variable between blocks.

In this paper, we present alternating asymmetric search range assignment (AASRA) to reduce the ME complexity in a stable ratio.

### 4.2 Alternating Asymmetric SR Assignment Algorithms

#### 4.2.1 AASRA for Bidirectional ME (AASRA-B)

B frames are coded using references from both past and future directions. Statistically, the two nearest (one past and one future) reference frames, as shown in Figure 4-1, are the most important to the prediction efficiency. Actually, in recent
implementations of high-throughput video encoders [39] [40], only these nearest references are searched to keep the complexity and memory bandwidth reasonable. Compared to P frames with only one reference direction and consequently only one nearest reference frame, an important observation from B frames is that the importance of each direction becomes relatively lower. Therefore, we try to reduce the total computational complexity by applying a weaker ME to one of the reference directions.

![Alternating asymmetric search range assignment for B frames.](image)

Figure 4-1 Alternating asymmetric search range assignment for B frames.

Since the complexity of ME is dependent to the size of search range, an asymmetric search range assignment (ASRA) method is first considered: to always use a large search range (SR.L) for one direction while assigning a small search range (SR.S) to the other. However, for high-motion video sequences that require a larger search range than SR.S, ASRA leads to inaccurate prediction in the direction with SR.S. This may result in a significant coding performance drop.
To overcome this drawback, we propose an alternating asymmetric search range assignment (AASRA) scheme. Unlike the fixed assignment of two search ranges to two directions as in ASRA, AASRA switches the use of SR.L and SR.S for past and future references once per block (MB or CTB), as shown in Figure 4-1. Regarding a given reference direction, if SR.L is used for block (N), then SR.S should be used for block (N+1), and block (N+2) switches back to SR.L. While the search center with SR.L can be either zero or the motion vector predictor (MVP) according to specific design considerations, the search center with SR.S should always be MVP.

![Figure 4-2 Comparison on motion vector tracking capabilities.](image)

Theoretically, AASRA for bidirectional ME (AASRA-B) has the following advantages. Firstly, the ME complexity is stable for each MB/CTB, which is important for ensuring the worst-case performance. When SR.L-to-SR.S ratio is large enough, the complexity reduction relative to applying SR.L to both directions can be close to 50%. This also reduces the coding complexity variation between B and P frames, which contributes to a better hardware utilization of real-time systems in coding P frames. Secondly, for each specific direction, an SR.L-based search is
always placed prior to an SR.S based one. While the former produces an accurate matching for a high motion, it also tends to provide the latter with a good search center. This is likely to result in another good matching even if SR.S is not large. Finally, in contrast to ASRA, AASRA-B attaches equal importance to both reference directions. If SR.L is also based on the MVP search center,

AASRA-B makes it possible for the motion vectors to capture a real motion larger than SR.L. As shown in Figure 4-2 (a), this is likely to be achieved from accumulated searches for two or more MBs/CTBs, similar to always performing SR.L searches. Figure 4-3 shows an example where a background horizontal motion of 27 integer pixels is captured with SR.L-SR.S set to 16-4. On the other hand, accumulating multiple SR.S searches cannot be as efficient (Figure 4-2 (b)).

The captured horizontal background motion is 27 integer pixels. The frame is encoded by SR.L-SR.S = 16-4.

Figure 4-3 Motion vector information encoded by SR.L-SR.S=16-8.

Figure 4-4 plots the trends of coded bit rates under various search ranges. These sequences are tested both on JM [50] and HM [51]. For AASRA-B, SR.S is set as a quarter of SR.L. This reduces complexity in terms of the number of search points by
Alternating Asymmetric Search Range Assignment for Motion Estimation

46.875\% (=\frac{1-(1/4)^2}{2}) compared to original full search (JM and HM) with SR = SR.L. In the meanwhile, the two curves for AASRA-B and reference software (JM or HM) are close to each other, which indicate only a small drop of coding performance.

Figure 4-4  Comparison between AASRA-B and original full search.
Note: JM is configured with a frame structure of IBBBP. HM is configured with hierarchical-B structure with GOP size of 8. For both JM and HM, 1 or 2 reference frames are used for P and B frames, respectively. QP = 32.

4.2.2 Applying AASRA in P Frames (AASRA-P)

Whereas AASRA-B is proposed for bidirectional ME, the similar idea of alternating search range assignment can also be applied in P frames with only one reference direction. AASRA for P frames (AASRA-P) starts with a large search range (SR.L) for the first block (MB or CTB) in a frame, and alternates to a smaller search range (SR.S) or back to SR.L once per block.

The process is illustrated in Figure 4-5 and is the same as the behavior of AASRA-B (Figure 4-1) in a single direction. Although the ME complexity for each block is variable with AASRA-P, the complexity for every block pair consisting of
two neighboring blocks is stable. Compared to original full search with SR = SR.L, AASRA-P reduces overall ME complexity in terms of the number of search points by
\[(1-(\text{SR.S}/\text{SR.L})^2)/2, \text{ near 50% when SR.S}^2 << \text{SR.L}^2.\] This is also the same as the reduction ratio of AASRA-B for B frames.

### 4.2.3 Combining AASRA-P and AASRA-B (AASRA-PB)

While AASRA-B and AASRA-P alternate large and small search ranges in the two dimensions of reference direction and MB/CTB index respectively, the two schemes can be combined to achieve further complexity reduction for bidirectional ME. Figure 4-6 shows the combined AASRA-PB scheme. Every two consecutive blocks (MB or CTB) are considered as a pair. With bidirectional search performed for each block, one block pair corresponds to 4 searches out of which 1 uses a large search range (SR.L) and the rest 3 use a small one (SR.S). Once per block pair the SR.L-based search rotates to a different quadrant of reference direction and the parity of block index, as shown in Figure 4-6. A complete rotation covering all the quadrants is finished in 4 block pairs and restarted from the next block. Compared to original full search with SR = SR.L, AASRA-PB reduces overall search points by
\[(3-3*(\text{SR.S}/\text{SR.L})^2)/4, \text{ which is over 70% when SR.S is set as a quarter of SR.L.}\]

In addition to higher complexity reduction for bidirectional search, another advantage of AASRA-PB is in balancing the complexity for P and B frames. In a coding workload containing both types of frames, if AASRA-B is already used for the B frames, applying AASRA-P in the P frames does not further reduce the worst-case ME complexity since the original complexity of P frames is already smaller than that of the B frames with AASRA-B. However, applying AASRA-P and AASRA-PB respectively in P and B frames can minimize both the average and worst-case complexity.
Alternating Asymmetric Search Range Assignment for Motion Estimation

Combined AASRA-P and AASRA-B.

Figure 4-6 SR.L rotation of AASRA-B.
4.2.4 Hardware Complexity Analysis

For hardware architecture with specific organization of processing elements (PE) and memory, the complexity may not be exactly proportional to the number of search points. We select the snake scan based architecture as an example to analyse the hardware complexity reduction achievable from the proposed algorithms. Snake scan [52] is a popular memory access method used for full-search ME.

As shown in Figure 4-7, it consists of five basic steps to update a shifter register array storing the reference block. After N clock cycles for pre-loading one NxN, the register array provides the PE with the necessary data for one search point per cycle. For a search window with \((2SR + 1)^2\) search points, the required number of processing cycles is:

\[
T_{SR} = (2SR + 1)^2 + N - 1 \quad (1)
\]

Assuming 1 reference frame is used for each prediction direction and NxN is the size of an MB/CTB, with the classic symmetric search range assignment, \(2T_{SR}\) clock cycles are required to complete the bidirectional search for each MB/CTB. Note that the snake scan method does not impose restrictions on search range, so it shouldn’t be a problem for one ME architecture to be designed as configurable to support multiple
search ranges. By using the same parallelism, the corresponding cycle count requirement for AASRA-B is equal to $T_{SR.L} + T_{SR.S}$. By setting $SR.L = SR$ and $SR.S = \lambda SR$, where $\lambda < 1$, the processing time reduction ratio by applying AASRA-B is equal to:

$$
\Delta c = 1 - \frac{T_{SR.L} + T_{SR.S}}{2T_{SR}}
= 1 - \frac{T_{SR} + T_{\lambda SR}}{2T_{SR}}
= 0.5 - \frac{(2\lambda SR + 1)^2 + N - 1}{2((2\lambda SR + 1)^2 + N - 1)}
\approx 0.5 - \frac{\lambda^2}{2}, \text{ when } SR^2 \gg N
$$

(2)

When $SR = 128$, $\lambda = 0.25$ and $N = 16$, the processing time reduction by applying AASRA-B with a snake scan architecture is more than 46%, approximating the reduction ratio of search points.

The processing time reduction ratio above can be regarded as equivalent to the hardware complexity reduction ratio, since the circuit area overhead involved by applying the AASRA schemes is negligible.

![Diagram](image-url)

**Figure 4-8** Additional control logic required to implement AASRA
Figure 4-8 shows the additional hardware required to implement AASRA. In snake scan, the search range parameter is mainly used for updating the address of the reference data memory. When the conventional scheme is used, the search range is always SR.L. When AASRA is used, search range should be dynamically configurable to be SR.L or SR.S. This requires an additional register for storing SR.S, a multiplexer for selection between SR.S and SR.L, and a corresponding status register which is updated upon the completion of every MB/CTB, as shown inside the dotted line in Figure 4-8. However, the consequent area increase is negligible compared to the typical size of an ME architecture (e.g. 492K-1152K for [53]).

The hardware complexity reduction ratio of AASRA-P is the same as AASRA-B. For AASRA-PB, the processing time is \( T_{SR.L} + 3T_{SR.S} \) for each MB/CTB pair. The corresponding time for fixed and symmetric search range assignment is \( 4T_{SR:L} \). Therefore, the processing time reduction by applying AASRA-PB is:

\[
\Delta C = 1 - \frac{T_{SR:L} + 3T_{SR:S}}{4T_{SR}} \\
= 1 - \frac{T_{SR} + 3T_{ASR}}{4T_{SR}} \\
= 0.75 - \frac{3((2\lambda SR + 1)^2 + N - 1)}{4((2\lambda SR + 1)^2 + N - 1)} \\
\approx 0.75 - \frac{3\lambda^2}{4}, \quad \text{when } SR^2 \gg N \tag{3}
\]

When \( SR = 128, \lambda = 0.25 \) and \( N = 16 \), the hardware complexity reduction is more than 70%.

To implement ME in hardware, not only the ME calculation component but also a memory hierarchy is required for reference data retrieval and feeding into the pipeline. Figure 4-9 shows a typical organization of the related components. The DRAM controller fetches the reference data from the external memory and stores them into an on-chip cache. The cache is usually shared by multiple components of
the encoder including integer ME (IME), fractional ME and motion compensation. The target of this work is to reduce the complexity of the IME calculation component. For snake scan based hardware, the reduction ratio has been defined in equations (1), (2) and (3). In the meanwhile, (1) also defines the data fetching clock cycles for IME calculation. So the bandwidth between the on-chip cache and IME can be reduced by the same ratio as IME complexity reduction.

![Figure 4-9 Typical organization of ME related components in video encoder.](image)

On the other hand, the mechanism to fetch data from the controller to the cache does not have to be changed. Correspondingly the bandwidth between DRAM controller and cache and between external DRAM and DRAM controller will not be influenced.

As shown in Fig. 11, when AASRA performs SR.S search, the SR.S search window is always smaller than and ensured to be contained by the original SR.L window. Sometimes when SR.L and SR.S are configured to be zero-centered and MVP-centered, respectively, the SR.S window with MVP-specified position may not be initially contained by the SR.L window. In this case, the SR.S window is moved to the nearest position where it can be contained. Thereby, even for SR.S search, the
original data fetching mechanism from DRAM controller can be used to obtain the whole SR.L window. Note that reducing the external bandwidth is not a target of our scheme and this is also difficult to achieve by AASRA because the search windows of neighboring blocks are already reused in the cache.

![Diagram of search range assignment](image)

**Figure 4-10** The SR.S window is always contained by the SR.L one.

To summarize, the AASRA schemes reduce the complexity of IME calculation and the bandwidth between on-chip cache and IME. Meanwhile it does not influence the external bandwidth and the data fetching mechanism from DRAM controller to the cache.

### 4.3 Combined with Existing ME Architectures

In addition to their application in full-search ME, the proposed AASRA schemes also have the adaptability to be combined with many existing fast algorithms and architectures to achieve an incremental reduction of complexity. We regard this a
Alternating Asymmetric Search Range Assignment for Motion Estimation

crucial feature in practice since numerous research results on ME design are already available. In this section, we select MB-parallel ME [53] and hierarchical ME [42][43] for case study and propose the necessary adaptations for combining AASRA with them. For some other representative architectures such as [40] that apply full search in a more straightforward style, it should be quite obvious that AASRA can work together with them smoothly.

4.3.1 AASRA Combined with MB-parallel ME

The MB-parallel data reuse (IMNPDR) [53] scheme was developed to reduce the bandwidth of on-chip memory, which can effectively save SRAM area and power dissipation especially for high-throughput video encoders. IMNPDR is to perform the ME of multiple MBs simultaneously so that the memory traffic for overlapped part of the search windows can be shared. In a typical configuration of IMNPDR for H.264/AVC 1080p encoding, search range is set as 32 when 4 MBs are processed in parallel.

One obstacle to apply AASRA-B with IMNPDR is that the MBs processed in parallel should share the same relative search center. In original IMNPDR, this is ensured by always performing zero-center ME. For AASRA-B, zero search center is applicable to the SR.L direction. But ME in the SR.S direction must follow a dynamic search center such as MVP to take advantage of the accurate search center provided by previous MBs, as explained in section 4.2.1.

To resolve this problem, for SR.S search, we define the search center of 4 horizontally neighboring MBs to be processed in parallel as shown in Figure 4-11, which is equal to the median value of left MV$_A$, upper right MV$_C$ and MV$_B$ that is the average of upper motion vectors MV$_{B_0}$, MV$_{B_1}$, MV$_{B_2}$, MV$_{B_3}$. The 4 MBs also share the same search range in the same reference direction, while the switches between SR.S and...
SR.L are performed once per 4 MBs. By doing so, IMNPDR can be applied with the
dynamic feature of SR.S search preserved.

<table>
<thead>
<tr>
<th>MV_A</th>
<th>MV_B0</th>
<th>MV_B1</th>
<th>MV_B2</th>
<th>MV_B3</th>
<th>MV_C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MB0</td>
<td>MB1</td>
<td>MB2</td>
<td>MB3</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{MV_B} = \text{Average} (\text{MV_B0, MV_B1, MV_B2, MV_B3})
\]

\[
\text{Search center} = \text{Median} (\text{MV_A, MV_B, MV_C})
\]

Figure 4-11  Search center decision for AASRA based IMNPDR.

Following snake scan, the cycle count IMNPDR requires for processing p MBs
in parallel is:

\[
T_{sr} = (2SR+1) \cdot (2SR+1+(p-1) \cdot N) + N-1 \quad (4)
\]

Here the additional cycle count relative to original snake scan equation (1) is
from the partial PE idle time for non-overlapped part of the search window. With SR
= 32, SR.L = SR, SR.S = 0.25SR, p = 4 and N = 16, the cycle count and complexity
reduction by applying AASRA-B on the basis of IMNPDR is approximately 43% by
substituting (4) into (2).

Since AASRA-P can be regarded as AASRA-B performed in a single reference
direction, it can be applied to IMNPDR in the same way as AASRA-B, achieving the
same complexity reduction ratio for P frames.

To apply AASRA-PB with IMNPDR, 4 neighboring MBs are taken as an MB
group which share the same search center as derived from the method shown in
Figure 4-11. Every two consecutive groups form a group pair. AASRA-PB can
thereby be realized by performing SR.L rotation once per group pair. With SR = 32,
SR.L = SR, SR.S = 0.25SR, p = 4 and N = 16, the complexity reduction of AASRA-PB on IMNPDR can be approximately 64% by substituting (4) into (3).

Consequently, IMNPDR can get 72% reduction, and by combining with AASRA-B and AASRA-PB, it can be further reduced by 43% and 64%, respectively.

4.3.2 AASRA Combined with Hierarchical ME

Hierarchical search [42][43] is an effective method to realize ME with a large search range. The PMRME architecture in [42] applies three search levels based on original (L0), 1:4-down-sampled (L1), and 1:16-down-sampled (L2) references to cover the search ranges of 8, 32 and 128, respectively. These levels are processed in parallel with dedicated circuits. While zero-center ME is performed in L1 and L2, MVP is used by L0 as the search center. Taking both search range and resolution into consideration, the three ME levels are very similar in computational complexity.

To combine the AASRA schemes with PMRME, the SR.L search is performed with a complete 3-level PMRME. Meanwhile, the SR.S search only uses L0 in PMRME which is originally based on the MVP search center. In this configuration, SR.L and SR.S can be regarded as 128 and 8, respectively.

<table>
<thead>
<tr>
<th>P Frame</th>
<th>PMRME</th>
<th>AASRA-B + PMRME</th>
<th>AASRA-P + PMRME</th>
<th>AASRA-PB + PMRME</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>x 1</td>
<td>L0</td>
<td>x 1</td>
<td>L0</td>
</tr>
<tr>
<td>L1</td>
<td>x 1</td>
<td>L1</td>
<td>x 0.5</td>
<td>L1</td>
</tr>
<tr>
<td>L2</td>
<td>x 1</td>
<td>L2</td>
<td>x 0.5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B Frame</th>
<th>PMRME</th>
<th>AASRA-B + PMRME</th>
<th>AASRA-P + PMRME</th>
<th>AASRA-PB + PMRME</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>x 2</td>
<td>L0</td>
<td>x 2</td>
<td>L0</td>
</tr>
<tr>
<td>L1</td>
<td>x 2</td>
<td>L1</td>
<td>x 1</td>
<td>L1</td>
</tr>
<tr>
<td>L2</td>
<td>x 2</td>
<td>L2</td>
<td>x 1</td>
<td>L2</td>
</tr>
</tbody>
</table>

Figure 4-12 Relative hardware parallelism requirement comparison
Figure 4-12 shows the relative hardware parallelism required to achieve the same processing throughput. Taking the original PMRME for P frames as a baseline, 1 time of parallelism is required on each level. By applying AASRA-P, L1 and L2 are only performed for the SR.L search once every two MBs, so the required parallelism for these two levels can be halved. This results in 33% reduction in overall complexity assuming the 3 levels of original PMRME have the same hardware cost. For B frames, the original PMRME requires 2 times of parallelism on each level for the 2 reference directions. With AASRA-B performing L1 and L2 searches for SR.L in only one reference direction, only 1 time of parallelism is required on each of these two levels, resulting in 33% reduction in overall complexity compared to original PMRME. For AASRA-PB, the required parallelism for L1 and L2 is further halved. Compared with original PMRME, this achieves 50% complexity reduction for B frames.

Consequently, PMRME can get 49.5% reduction, and by combining with AASRA-B and AASRA-PB, it can be further reduced by 33% and 50%, respectively.

### 4.4 Experiments and Evaluation

#### 4.4.1 Equivalent Fixed Search Range for AASRA

The contribution of the search range parameter to coding efficiency is highly dependent to video contents. Large search range does not contribute to the compression ratio of still or low-motion sequences. Even for high-motion sequences, the coding efficiency increase with search range stops at a content-dependent point related to motion distances. After this specific point, further increase of search range can even lead to a slight decrease of coding efficiency due to the increased chances to obtain large fake motion vectors that do not reflect the real motion. As a result, it can
be difficult to understand the effectiveness of improved search range assignment by simply observing isolated search range points.

In section 4.2.1, we try to overcome the above issue by plotting the trend of bit rate decrease with search range increase in Figure 4-4. The coding efficiency (bit rate) difference from applying the proposed schemes can thereby be visible as the distance between two curves covering a wide interval of search ranges. However, there are still several remaining problems: 1) the video quality (PSNR) difference has not been taken into consideration; 2) only a single QP value can be tested at one time; 3) complexity is difficult to measure from the figure.

![Figure 4-13 Evaluation of equivalent fixed search range.](image)

To solve problems 1) and 2), we plot Figure 4-13 based on the Bjontegaard Difference (BD) measure [54]. The rate distortion (RD) performance of JM at SR = 8 is used as a baseline point relative to which the BD-rate of larger search range points are calculated with QPs set as 22, 27, 32 and 37. These points are then connected one
by one to form a curve for JM. Another curve for AASRA (specifically AASRA-B in Figure 4-13) is derived using the same method and same baseline point.

To solve problem 3), each test point of AASRA-B is horizontally projected to the BD-rate axis and the connection point of the projection line and the JM curve is taken as an equivalent fixed search range (EFSR) point. EFSR estimates the search range required by JM to achieve the same RD performance of the corresponding test point of AASRA-B. Thereby the factor of coding efficiency difference can be fairly removed from comparison. The complexity reduction ratio can be calculated using equation (2) for AASRA-B and AASRA-P, or (3) for AASRA-PB, by substituting search range with EFSR.

As shown in Figure 4-13, for AASRA-B with SR.S equal to a quarter of SR.L, EFSR is 15, 27, 59 or 122, when SR.L is 16, 32, 64 or 128. The corresponding complexity reduction is 38.52%, 25.26%, 37.36% or 41.45%. Table 4-1 shows the results of EFSR on more sequences and test configurations. The average complexity reduction by applying AASRA-B, AASRA-P and AASRA-PB is 34.2%, 28.9%, and 43.1%, respectively, with no coding efficiency degradation.

![Figure 4-14](image)

*Figure 4-14  Evaluation of EFSR on the 1080p Pedestrian sequence.*
It should be noted that EFSR has some limitations. EFSR is not so meaningful when BD-rate only slightly decreases or even increases from one test point to the next. So the proposed evaluation method can only be used for high-motion sequences over a limited interval of search ranges. For example, in Table 4-1, we are not able to appropriately calculate the EFSR of AASRA-P for SR.L set to 16, due to the irregular trend of BD-rate at these points (as plotted in Figure 4-14). Moreover, derivation of EFSR requires extensive computation. For each sequence over each configuration, 32 encoding iterations are needed for the combinations of 4 QPs, 4 search range points and 2 curves.
### Table 4-1 Complexity comparison by EFSR

<table>
<thead>
<tr>
<th>Avg. adjusted CRR overall</th>
<th>Avg. adjusted CRR</th>
<th>Pedestrian</th>
<th>Avg. adjusted CRR</th>
<th>Tractor</th>
<th>Avg. adjusted CRR</th>
<th>Woman</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. adjusted CRR</td>
<td>34.2%</td>
<td>14</td>
<td>32.8</td>
<td>-16.4</td>
<td>12832</td>
<td>64.8</td>
<td>SRL-SR-S</td>
</tr>
<tr>
<td>BD-rate</td>
<td>114</td>
<td>25</td>
<td>26</td>
<td>86</td>
<td>120</td>
<td>282</td>
<td></td>
</tr>
<tr>
<td>Adjusted CRR*</td>
<td>27.1%</td>
<td>-0.40%</td>
<td>33.3%</td>
<td>14.0%</td>
<td>27.1%</td>
<td>34.4%</td>
<td>AASRA-B</td>
</tr>
<tr>
<td>BD-rate</td>
<td>121</td>
<td>13.15%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AASRA-P</td>
</tr>
<tr>
<td>Adjusted CRR*</td>
<td>47.1%</td>
<td>-11.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AASRA-PB</td>
</tr>
<tr>
<td>BD-rate</td>
<td>104</td>
<td>53.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*: Complexity reduction ratio (CRR) relative to full-search JM, which has been adjusted to compensate the RD performance drop by using EFSR in equations (2) and (3).
4.4.2 Rate-Distortion Based Evaluation

As explained in section 4.4.1, the EFSR-based evaluation method has some limitations despite of its usefulness especially for high-motion sequences. In this section, as a compliment to EFSR, we present a more conventional rate-distortion (RD) based evaluation for AASRA on a wider set of video sequences and test configurations.

The experiments are conducted on the basis of JM 17.2 and HM 7.0 to make comparisons between before and after the proposed AASRA schemes are applied on JM, IMNPDR, PMRME, and HM. AASRA is configured to use an SR.L equal to the fixed search range used by the original ME algorithms. SR.S is smaller. The same algorithm configurations are used as in sections 4.2, 4.3.1 and 4.3.2. Sequences with resolutions ranging from 832x480 to 3840x2160 are tested. The test sequences are classified into two categories of high motion and median motion. All test sequences are encoded using four QPs including 22, 27, 32 and 37, based on which BD-PSNR and BD-rate are calculated to reflect the overall difference of RD performance.

<table>
<thead>
<tr>
<th>Test conditions</th>
<th>H.264/AVC</th>
<th>HEVC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test model</td>
<td>JM 17.2</td>
<td>HM 7.0</td>
</tr>
<tr>
<td>GOP structure</td>
<td>IBBBBP</td>
<td>Hierarchical B</td>
</tr>
<tr>
<td>GOP size</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Search range</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>MB/CTB size</td>
<td>16x16</td>
<td>64x64</td>
</tr>
<tr>
<td>Reference number of B</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Reference number of P</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>QP</td>
<td>22, 27, 32, 37</td>
<td></td>
</tr>
</tbody>
</table>
Table 4-2 summarizes the test conditions. For full-search JM, search range is set as 128. SR.L-SR.S for the AASRA schemes is set as 128-32. The SR.L-SR.S parameters for AASRA based on IMNPDR and PMRME are 32-8 and 128-8, respectively. Search range for the HM test sequences is set as 64 as recommended by HEVC common test conditions [57]. Correspondingly, SR.L-SR.S of AASRA implemented on HM is set to 64-16. Coding structure is IBBBP for JM. HM is configured with random access (hierarchical-B, GOP 8) structure. For both JM and HM, numbers of reference frames used for B and P frames are set as 2 and 1, respectively.

Table 4-3 and Table 4-5 show the comparison between the AASRA schemes and the original full-search ME algorithms on JM and HM, respectively. To demonstrate the effectiveness of AASRA for high-motion video, some sequences are tested with an additional configuration in which 1 out of every 2 frames is skipped. Table 4-4 shows the experiments on different ratios between SR.L and SR.S, which indicates 4:1 can be a good trade-off between complexity and coding efficiency. Figure 4-16 shows a subjective visual quality comparison. Due to large irregular motion in the selected area, the motion vector patterns coded with and without AASRA are significantly different from each other. In the meanwhile, subjective quality is almost the same, which fits PSNR well. Table 4-6 shows the results from applying AASRA on the basis of IMNPDR and PMRME.

As shown in Table 4-3, Table 4-5, and Table 4-6, the RD performance drop for median-motion sequences is near negligible, since the motion of these sequences barely exceeds the search range. For high-motion sequences, the average RD performance drop from applying AASRA-B is 0.18% to 0.91% in terms of BD-rate increase. This can be regarded as small considering the 33% to 46% complexity reduction achieved for B frames. The average BD-rate increase from simultaneously
applying AASRA-P and AASRA-PB ranges from 0.42% to 2.14%, which is reasonably larger due to the more aggressive reduction in complexity (50% to 70% for B frames).

(a) 1st B frame in the IBBBP Tractor sequence.

(b) Encoded by full-search JM (PSNR 43.190). The left picture shows the MV distribution.

(c) Encoded by AASRA-PB (PSNR 43.185). The left picture shows the MV distribution.

Figure 4-16 Subjective quality comparison.
Table 4-3 RD performance of AASRA on full-search JM.

<table>
<thead>
<tr>
<th>Sequences</th>
<th>AASRA-B</th>
<th>AASRA-PB/P**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BD-PSNR (dB)</td>
<td>BD-rate (%)</td>
</tr>
<tr>
<td>High Motion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3840x2160 ParkJoy</td>
<td>0.00</td>
<td>-0.04</td>
</tr>
<tr>
<td>1920x1080 Tractor*</td>
<td>-0.03</td>
<td>0.69</td>
</tr>
<tr>
<td>1920x1080 Tractor</td>
<td>-0.01</td>
<td>0.22</td>
</tr>
<tr>
<td>1920x1080 Pedestrian*</td>
<td>-0.02</td>
<td>0.67</td>
</tr>
<tr>
<td>1920x1080 Pedestrian</td>
<td>-0.01</td>
<td>0.38</td>
</tr>
<tr>
<td>1920x1024 Woman*</td>
<td>-0.12</td>
<td>2.96</td>
</tr>
<tr>
<td>1920x1024 Woman</td>
<td>-0.03</td>
<td>0.87</td>
</tr>
<tr>
<td>1920x1024 Crosswalk*</td>
<td>-0.02</td>
<td>0.51</td>
</tr>
<tr>
<td>1920x1024 Crosswalk</td>
<td>-0.0</td>
<td>0.16</td>
</tr>
<tr>
<td>Average</td>
<td>-0.03</td>
<td>0.71</td>
</tr>
<tr>
<td>Median Motion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3840x2160 CrowdRun</td>
<td>0.00</td>
<td>-0.13</td>
</tr>
<tr>
<td>3840x2160 InToTree</td>
<td>0.00</td>
<td>-0.02</td>
</tr>
<tr>
<td>1920x1080 Station</td>
<td>0.00</td>
<td>-0.06</td>
</tr>
<tr>
<td>1920x1024 Whale</td>
<td>0.01</td>
<td>-0.21</td>
</tr>
<tr>
<td>1920x1024 Flamingoes</td>
<td>0.00</td>
<td>-0.02</td>
</tr>
<tr>
<td>1920x1024 Marching</td>
<td>0.01</td>
<td>-0.16</td>
</tr>
<tr>
<td>1920x1080 RushHour</td>
<td>0.01</td>
<td>-0.48</td>
</tr>
<tr>
<td>1280x720 Mobcal</td>
<td>0.01</td>
<td>-0.26</td>
</tr>
<tr>
<td>1280x720 City</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>Average</td>
<td>0.00</td>
<td>0.14</td>
</tr>
<tr>
<td>CRR*** (P frames)</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>CRR (B frames)</td>
<td>46%</td>
<td></td>
</tr>
</tbody>
</table>

*: Sequences coded with 1-frame skip.

**: Applying AASRA-PB for B frames and AASRA-P for P frames

***: Complexity Reduction Ratio.
Table 4-4 RD Performance of different SR.L-SR.S ratio.

<table>
<thead>
<tr>
<th>Sequences</th>
<th>SR.L-SR.S</th>
<th>AASRA-B</th>
<th>AASRA-PB/*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BD-PSNDR</td>
<td>BD-rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(dB)</td>
<td>(%)</td>
</tr>
<tr>
<td>Tractor</td>
<td>128-64</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>128-32</td>
<td>-0.01</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>128-16</td>
<td>-0.02</td>
<td>0.31</td>
</tr>
<tr>
<td>Woman</td>
<td>128-64</td>
<td>-0.01</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>128-32</td>
<td>-0.03</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>128-16</td>
<td>-0.04</td>
<td>1.03</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>128-64</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>128-32</td>
<td>-0.01</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>128-16</td>
<td>-0.02</td>
<td>0.76</td>
</tr>
<tr>
<td>Average</td>
<td>128-64</td>
<td>0.00</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>128-32</td>
<td>-0.02</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>128-16</td>
<td>-0.03</td>
<td>0.70</td>
</tr>
<tr>
<td>CRR**</td>
<td>128-64</td>
<td></td>
<td>37%</td>
</tr>
<tr>
<td></td>
<td>128-32</td>
<td></td>
<td>46%</td>
</tr>
<tr>
<td></td>
<td>128-16</td>
<td></td>
<td>49%</td>
</tr>
</tbody>
</table>

*: Applying AASRA-PB for B frames and AASRA-P for P frames.

**: Complexity Reduction Ratio.
Table 4-5 RD performance of AASRA on full-search HM

<table>
<thead>
<tr>
<th>Sequences</th>
<th>AASRA-B</th>
<th>AASRA-PB/P*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BD-PSNR</td>
<td>BD-rate</td>
</tr>
<tr>
<td></td>
<td>(dB)</td>
<td>(%)</td>
</tr>
<tr>
<td><strong>High Motion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1920x1080 RaceHorses</td>
<td>0.00</td>
<td>0.15</td>
</tr>
<tr>
<td>1920x1080 Kimono</td>
<td>-0.01</td>
<td>0.24</td>
</tr>
<tr>
<td>832x480 BasketballDrill</td>
<td>-0.01</td>
<td>0.22</td>
</tr>
<tr>
<td>2560x1600 Traffic</td>
<td>0.00</td>
<td>0.09</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>-0.01</td>
<td>0.18</td>
</tr>
<tr>
<td><strong>Median Motion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2560x1600 PeopleOnStreet</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>1920x1080 Cactus</td>
<td>0.00</td>
<td>-0.02</td>
</tr>
<tr>
<td>1280x720 Vidyo1</td>
<td>0.00</td>
<td>0.09</td>
</tr>
<tr>
<td>1280x720 Vidyo4</td>
<td>0.00</td>
<td>-0.06</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>CRR** (P frames)</td>
<td>0%</td>
<td>46%</td>
</tr>
<tr>
<td>CRR (B frames)</td>
<td>46%</td>
<td>70%</td>
</tr>
</tbody>
</table>

*: Applying AASRA-PB for B frames and AASRA-P for P frames.

**: Complexity Reduction Ratio.
Table 4-6: RD Performance of AASRA on IMNPDR and PMRME

<table>
<thead>
<tr>
<th>CRR ** (P frames)</th>
<th>CRR (B frames)</th>
<th>Median motion</th>
<th>High motion</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>3840x2160 CrowdRun</td>
<td>1280x720 Motel</td>
<td>1920x1080 Station</td>
<td>1920x1024 Crosswalk</td>
<td>1280x720 Mobcal</td>
</tr>
<tr>
<td>3840x2160 InToTree</td>
<td>1920x1080 Pedestrian</td>
<td>1920x1024 Tractor</td>
<td>1920x1080 Woman</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRR ** (P frames)</th>
<th>CRR (B frames)</th>
<th>Median motion</th>
<th>High motion</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>3840x2160 CrowdRun</td>
<td>1280x720 Motel</td>
<td>1920x1080 Station</td>
<td>1920x1024 Crosswalk</td>
<td>1280x720 Mobcal</td>
</tr>
<tr>
<td>3840x2160 InToTree</td>
<td>1920x1080 Pedestrian</td>
<td>1920x1024 Tractor</td>
<td>1920x1080 Woman</td>
<td></td>
</tr>
</tbody>
</table>

* CRR: Complexity Reduction Ratio.

*: Applying AASRA-PB for B frames and AASRA-P for P frames.

**: Complexity Reduction Ratio.
4.5 Summary

To reduce ME complexity in a stable ratio, we propose three alternating asymmetric search range assignment (AASRA) schemes including AASRA-B, AASRA-P and AASRAPB.

In section 4.1, recent works on motion estimation are introduced including new search patterns and modified full search.

In section 4.2, three schemes including AASRA-B, AASRAP and AASRA-PB are proposed. AASRA-B uses a large and a small search ranges, respectively, for the two reference directions of bidirectional ME. The assignment of these two search ranges alternates between past and future references for each MB/CTB, enabling ME in both directions to track high motions while achieving over 46% saving of complexity. AASRA-P extends the application of AASRA to P, achieving the same complexity reduction ratio as AASRA-B. AASRA-PB combines the features of AASRA-B and AASRA-P to obtain over 70% complexity reduction for B frames with small coding performance drop.

In section 4.3, we further propose the necessary adaptations to apply AASRA on the basis of MB-parallel ME and hierarchical ME. This results in 33% to 64% incremental reduction of complexity.

In section 4.4, according to the conventional RD-based evaluation, AASRA-B, AASRA-P and AASRA-PB reduce ME complexity by 46%, 46%, and 70%, respectively, with small coding performance drop. To more appropriately evaluate the effectiveness of the proposed schemes for high-motion sequences, we also propose an equivalent fixed search range (EFSR) metric to compensate the coding efficiency degradation in the complexity comparison. After the compensation, AASRA still shows 28.9% to 43.1% complexity reduction.
5 Architectures Implemented in Video Decoder Chip

Inter prediction techniques mentioned in this dissertation including the motion compensation and parameter decoding have been implemented in video decoder chips. This chapter analyses the features of these components in a whole video decoder chip.

5.1 Implemented Chips

The cache based motion compensation is implemented in the 2Gpixels/s 7680x4320@60fps H.264/AVC HP/MVC video decoder chip [58] and 530Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip [15] [33]. Micrograph of the test chips are shown in Figure 5-1 and Figure 5-2. For the 8Kx4K chip, the die size is 4x4mm\(^2\) including a 64-bit DDR2 PHY, DLL, PLLs, and the digital core containing 1338K logic gates and 79.9KB on-chip memory. For the 4Kx2K chip, the die size is 4x4mm\(^2\) including DDR PHY, DLL, PLL, and the decoder core that contains 662K logic gates and 59.6KB on-chip memory.

The joint parameter decoder which includes the motion data calculation, boundary strength calculation, and intra prediction mode calculation is implemented in 2Gpixels/s H.264/AVC HP/MVC video decoder chip [58], 530Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip [15][33], and 1080p@60fps multi-standard video decoder chip [7].
Architectures Implemented in Video Decoder Chip

Figure 5-1  Chip micrograph of 8kx4k video decoder chip [58].

Figure 5-2  Chip micrograph of 4kx2k video decoder [15].
5.2 Features of Inter Prediction Components in Decoder Chip

Figure 5-3 shows the top-level block-diagram of 4kx2k@60fps video decoder. The blocks with shadow are the inter prediction related components. The logic gate for the JPDEC, MC cache, and MC interpolation is 37.2k, 37.6k, and 71.2k, respectively.

Table 5-1 shows the pipeline utilization of major components of the main decoder for various frame types. The result is based on the sequence IntoTree.264 with IPBB GOP structure, and QP24, and running at 166MHz.

Table 5-1 Pipeline utilization of major components [33]

<table>
<thead>
<tr>
<th>Component</th>
<th>Cycles/MB</th>
<th>Average pipeline utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>I frames</td>
</tr>
<tr>
<td>IT</td>
<td>64</td>
<td>91.1</td>
</tr>
<tr>
<td>JPDEC*</td>
<td>64</td>
<td>91.1</td>
</tr>
<tr>
<td>Intra</td>
<td>0~66</td>
<td>93.9</td>
</tr>
<tr>
<td>MC Interpolation *</td>
<td>0~65</td>
<td>0</td>
</tr>
<tr>
<td>Rec.</td>
<td>64</td>
<td>91.1</td>
</tr>
<tr>
<td>DBF</td>
<td>64</td>
<td>91.1</td>
</tr>
<tr>
<td>LFRC Coding</td>
<td>64</td>
<td>91.1</td>
</tr>
<tr>
<td>LFRC Restore</td>
<td>3 cycles/partition</td>
<td>0</td>
</tr>
</tbody>
</table>

*: Inter-frame coding related components.
Figure 5-3  Top-level block-diagram of 4kx2k@60fps video decoder [33].
For the MC interpolation component, the worst-case of throughput is 65 cycles/MB, when considering the maximum MV number and bi-prediction limits on high levels. As described in section 2.6.1, most of the MBs require 56 cycles, and the average processing is 48 cycles/MB. The speed requirement in the whole pipelined 4Kx2K decoder is 64 cycles/MB. The average speed of the proposed interpolation can meet the requirement. The utilization of interpolation is not so high, since the data throughput for motion compensation varies significantly according to the sequence features and the architecture should therefore be over-designed for the worst cases.

For the JPDEC component, by applying dual mode pipeline with simplified control modes, a constant 64 cycles/MB throughput can be obtained which can meet the speed requirement of the whole decoder. Moreover, the component can achieve high pipeline utilization from 85.5% to 93.9%. The utilization is lower for B frames, because the DRAM bandwidth instead of core performance becomes the system bottleneck during B frame decoding.
6 Conclusion

In this dissertation, efficient algorithms and architectures are proposed to reduce the computational complexity and memory bandwidth of inter prediction, and consequently decrease the area cost and memory power consumption of video codec VLSI.

Firstly, cache based motion compensation (MC) architecture is proposed to support real-time 4kx2k@60fps decoding at less than 166MHz, with 37.2k logic gates. A high-performance interpolator based on horizontal-vertical expansion and luma-chroma parallelism (HVE-LCP) is proposed to efficiently increase the processing throughput to at least over 4 times as the previous designs. Then, an efficient cache memory organization scheme (4Sx4) is adopted to improve the on-chip memory utilization, which contributes to memory area saving of 25% and memory power saving of 39%~49%. By employing a Split Task Queue (STQ) architecture, the cache system is 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%.

Secondly, VLSI architecture of a joint parameter decoder is proposed to realize the calculation of motion vector (MV), intra prediction mode (IPM) and boundary strength (BS). For this architecture, a dual-mode pipeline with simplified control modes is designed to increase system throughput and reduce hardware cost. Then, three-step bandwidth reduction strategy is applied. The data which includes the motion information for the co-located picture and the last decoded line, is pre-processed before being stored to DRAM. A partition based storage format is
applied to con- dense the MB level data, while variable length coding based compression method is utilized to reduce the data size in each partition. Experimental results show our design is capable of real-time 3840x2160@60fps decoding at less than 166MHz, with 37.2k logic gates. Meanwhile, by applying the proposed scheme, 85%~98% bandwidth saving is achieved, compared with storing the original information for every 4x4 block to DRAM.

Finally, alternating asymmetric search range assignment (AASRA) schemes for motion estimation is proposed which includes AASRA-B, AASRA-P and AASRA-PB. AASRA-B uses a large and a small search ranges, respectively, for the two reference directions of bidirectional ME. The assignment of these two search ranges alternates between past and future references for each MB/CTB, enabling ME in both directions to track high motions with reduced complexity. AASRA-P extends the application of AASRA to P frames, and AASRA-PB combines the features of AASRA-B and AASRA-P to obtain even higher complexity saving. AASRA-B, AASRA-P and AASRA-PB reduce ME complexity by 46%, 46% and 70%, respectively, with small coding performance drop. Then, we propose an equivalent fixed search range (EFSR) metric to compensate the coding efficiency degradation in the complexity comparison. After the compensation, AASRA still shows 28.9% to 43.1% complexity reduction. We also demonstrate AASRA’s flexibility to be combined with the state-of-the-art ME architectures including MB-parallel ME and hierarchical ME.

The cache based motion compensation is implemented in the 2Gpixels/s H.264/AVC HP/MVC video decoder chip and 530Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip. The joint parameter decoder is implemented in 2Gpixels/s H.264/AVC HP/MVC video decoder chip, 530Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip, and 1080p@60fps multi-standard video decoder chip.
Acknowledgement

First and foremost, I would like to gratefully and sincerely thank Professor Satoshi Goto of Waseda University for his constant encouragement, guidance, and support during my research. Professor Goto has wonderful personal fascination, and really gives me a great guide in my growth. I also express my appreciation to Professor Peilin Liu of Shanghai Jiao Tong University for the continuous encouragement and support throughout my work. I would also like to thank Professor Kimura and Professor Katto for giving me precious advices and continuous encouragement during my research.

I would like to appreciate Mr. Dajing Zhou of Waseda University. He gave me direct guidance and tremendous help during my research. I also thank Mr. Gang He, Mr. Xun, Ms. Muchen Li, and Mr. Wei Fei for working together with me on the topic of video chip design. And also the other students in Goto-Lab, they give me a lot of help both in my research and life. With these people’s help, I can enjoy the research life in Waseda University.

I also express my sincere gratitude to Japan Society of the Promotion of Science, for the support to my research. I also thank the support from Waseda University Ambient SoC Global COE Program of MEXT, Japan, and from the CREST project of Japan Science and Technology Agency.

Finally, an honorable mention goes to my family for their understandings and supports.
References


Publications

Journal Papers


Publications


International Conference Papers


