Unified Parameter Decoder Architecture for H.265/HEVC Motion Vector and Boundary Strength Decoding

Shihao WANG (a), Nonmember, Dajiang ZHOU †, Member, Jianbin ZHOU †, Nonmember, Takeshi YOSHIMURA †, and Satoshi GOTO †, Fellows

SUMMARY In this paper, VLSI architecture design of unified motion vector (MV) and boundary strength (BS) parameter decoder (PDec) for 8K UHDTV HEVC decoder is presented. The adoption of new coding tools in PDec, such as Advanced Motion Vector Prediction (AMVP), increases the VLSI hardware realization overhead and memory bandwidth requirement, especially for 8K UHDTV application. We propose four techniques for these challenges. Firstly, this work unifies MV and BS parameter decoders for line buffer memory sharing. Secondly, to support high throughput, we propose the top-level CU-adaptive pipeline scheme by trading off between implementation complexity and performance. Thirdly, PDec process engine with optimizations is adopted for 43.2k area reduction. Finally, PU-based coding scheme is proposed for 30% DRAM bandwidth reduction. In 90 nm process, our design costs 93.3k logic gates with 23.0 kB memory sharing. The proposed architecture can support real-time decoding for 7680x4320@60fps application at 249 MHz in the worst case.

key words: UHDTV, H.265/HEVC, parameter decoder, motion vector, boundary strength

1. Introduction

Nowadays Ultra High Definition Television (UHDTV) has been a hot topic because of the better visual experience it provides. Up to 8K UHDTV video format has been regarded as the high-end application in the near future. At the same time, the new High Efficiency Video Coding (H.265/HEVC) video coding standard has been standardized in January 2013. HEVC, as a successor to H.264, can achieve double video compression rate while guaranteeing the equivalent compression quality [1], [2]. Under this background, video decoder supporting both 8K UHDTV and HEVC is required by the market. However, in order to support real-time decoding ability, VLSI implementation for decoder is challenged by two issues. Firstly, since high video resolution directly increases the burden on system throughput because of the huge data volume, higher throughput requirement is needed. Assuming the clock frequency is 250 MHz, for 7680x4320@60fps, a throughput of at least 8 pixel/cycle has to be achieved. On the other hand, new HEVC standard introduces complicated coding tools in MV and BS calculation to achieve better compression performance, such as Advanced Motion Vector Prediction (AMVP) mode and merge mode. Flexible quad-tree CU structure is also introduced in HEVC. All these new coding tools lead to challenges on memory bandwidth requirement, data dependency and throughput requirement for high-performance VLSI implementation [3].

VLSI architecture of unified MV and BS PDec is discussed by following reasons. 1) As shown in Fig. 1, MV and BS decoder play the obligatory roles for an integrated video decoder system. 2) Original syntaxes from CABAC and BS decoder play the obligatory roles for an integrated video decoder system. 3) The pre-process for decoding syntaxes into parameters has less dependency with core calculation in MC and DBF. Further, BS calculation relies on the result of current block’s MV, which inspires us to build unified PDec architecture to share on-chip memory without much extra control overhead.

Several parameter decoder approaches are reported previously. In [4] an approach for MPEG2 is achieved. For H.264/AVC, Xu, et al. [5] first shows a solution for QCIF format. Higher throughput is also achieved in [6]–[8]. However, these realizations have much room for optimization. The fastest throughput among them is 260 cycles/MB, which is far slower than the throughput requirement for 8K UHDTV. In Zhou et al.’s work [9], a joint parameter decoder with fixed pipeline granularity for 4K@60fps UHDTV application is proposed for MV, BS and intra prediction mode. Considering this H.264/AVC approach is not applicable for HEVC and the technique can’t support high throughput requirement for 8K video, we can’t directly inherit [9] for
HEVC solution. Except the VLSI works mentioned above, FPGA-based implementation in [10] is also reported. However, FPGA based work is too difficult to be extended for 8K UHDTV application. Meanwhile, many previous works have been done on the motion compensation architecture for H.264/AVC and H.265/HEVC, such as [11]–[13]. However, few of them involves the realization for MV production, which is a tough work in HEVC for hardware implementation.

In this paper a novel VLSI PDec architecture is proposed with four contributions.

1) The unified architecture for MV and BS parameter decoder is proposed to achieve memory sharing.

2) The CU-adaptive pipeline is proposed to support high throughput of up to 8K application with reasonable implementation complexity.

3) The proposed Index-mapping and resource reuse schemes are introduced for irregular process algorithm to achieve 43.2 k logic gates reduction.

4) Memory organization is well designed and PU-based coding scheme for co-located storage can reduce 30% DRAM bandwidth requirement.

In total, our proposed parameter decoder architecture support real-time decoding for 7680x4320@60fps videos at 249 MHz clock frequency in worst case.

The rest of the paper is organized as followings. Section 2 presents an overview for PDec in HEVC. Section 3 shows PDec pipeline design in detail. In Sect. 4, both spatial and temporal storage with optimized schemes are discussed in detail. Implementation results are shown in Sect. 5. Finally, we conclude the whole paper in Sect. 6.

2. Overview of PDec in HEVC

In this section the overview of parameter decoder is discussed in detail. Our proposed unified parameter decoder contains two major parts, motion vector calculation and boundary strength calculation, as shown in Fig. 1. The calculation for intra prediction mode is excluded from the proposed PDec for following reasons. Firstly, in HEVC standard, decoding syntaxes of transform units in CABAC relies on the result of intra prediction mode for current coding unit. If intra part is encapsulated in PDec, a long feedback path inside the whole decoder is generated, leading to performance degradation. Secondly, algorithm for intra prediction is simpler than MV’s and it doesn’t need the reference data which is locates in different coding tree unit (CTU) rows. Therefore, the overhead for adding intra part into CABAC is negligible compared to MV and BS calculation. The rest of this chapter gives a brief overview for MV and BS calculation in HEVC.

2.1 Motion Vector Calculation

The process of calculating motion vector is to decode syntax elements into motion parameters, which can be directly used by the following motion compensation module. A block’s motion parameters have high possibility to be similar to spatial or temporal neighboring. In HEVC, irregular coding algorithm is employed to eliminate such kinds of redundancy for compression efficiency.

Advanced Motion Vector Prediction (AMVP) mode and merge prediction mode are employed by HEVC for coding MV parameters. Both of them require prediction parameters of five spatial neighboring blocks and two temporal co-located blocks as input, as depicted in Fig. 2. Besides, each of the two prediction modes owns its unique algorithm and resource sharing between them is limited. In AMVP, all seven reference blocks are categorized into three regions, A, B and Col regions. Each region will produce at most one candidate so that a list of at most three candidates can be constructed. After removing the identical ones in the list, the final MV will be selected by the syntax mvp_idx_flag. In merge mode, motion parameter decision starts with constructing merge candidate list. Firstly, valid blocks are pushed into the list in the order of B0, A0, A1, B1, B2 and Col. Then the combined candidates will be assembled with the content of candidate list and added into list with the valid candidates in the first step. Finally, zero candidates with different reference frames are produced if the list is not full. After the list is constructed, merge MV result is chosen by the merge_idx from the list. Note that in each mode, the scaling operations will be processed when there is a difference between reference frame of current block and that of reference block [2].

Scaling calculator is an operation frequently used in AMVP mode. Figure 3 shows two cases where scaling calculator is utilized. In the figure tb (td) indicates the POC difference between current (neighbor) picture and its reference picture. When tb and td are not the same, mvpNeighbor can’t be directly used as the prediction of motion vector and
it must be scaled. Therefore, mvScaled should be deduced by following equations defined in HEVC specification [2].

\[
\begin{align*}
tb &= \text{Clip}3(-128, 127, \text{Diff PicOrderCnt(curr_pic, curr_ref)}) \\
td &= \text{Clip}3(-128, 127, \text{Diff PicOrderCnt(neighbor_pic, neighbor_ref)}) \\
tx &= (16384 + (\text{Abs}(td) >> 1)) / td \\
distScaleFactor &= \text{Clip}3(-4096, 4095, (tb \times tx + 32) >> 6) \\
mvScaled &= \text{Clip}3(-32768, 32767, \text{Sign}(\text{distScaleFactor} \times \text{meNeighbor}) \\
&\times ((\text{Abs}(\text{distScaleFactor} \times \text{meNeighbor}) + 127) >> 8))
\end{align*}
\]

2.2 Boundary Strength Calculation

Boundary strength is used in deblocking filter for filter selection. The calculation of BS can be divided into two steps. The first step is to prepare necessary data. Generally, we have to fetch the MV parameters of current block and all the adjacent neighboring blocks in the left and top. Then in the second step, specific algorithm is used to produce the BS result by comparing MV parameters between current block and neighboring blocks.

BS calculation will be executed on all the prediction unit (PU) and transform unit (TU) edges at 8x8 block grid. Let P and Q be the two blocks beside a certain edge. If P or Q is intra prediction, BS is equal to 2. Otherwise, PU and TU edges have different algorithms to produce BS value. The algorithm for TU edges is quite simple, which just check whether P or Q has non-zero coefficients. In contrast, the algorithm for PU edges are much more complex. Almost all the motion information has to be compared between P and Q blocks, such as the number of reference frames, the number of MV and the MV difference [2].

3. Unified CU-adaptive Pipeline

3.1 Pipeline Analysis

In order to support high throughput requirement, the pipeline design involves three aspects, pipeline granularity, pipeline processing stages and cycle resource allocation.

Pipeline granularity should be well designed for trading off between performance and implementation complexity. Pipeline granularity in PDec is a set of pixels processed by the processing element simultaneously, which means the pixels in granularity should be homogeneous. In HEVC, pixels in one coding unit share the same prediction mode (inter or intra), pixels in each PU share one MV. On the other hand, 4-pixel-length edges on 8x8 grid share one BS for deblocking filter. By combining these two processes, it’s better to define the pipeline granularity as 8x8 blocks or larger. In previous works, fixed pipeline granularity is adopted. Though pipeline controller for that is simple, the throughput will be decreased in HEVC. For example, in [6], fixed 4x4 or 8x8 pipeline block is designed. However, HEVC supports maximum 64x64 block size. If fixed 4x4 pipeline granularity is used, large redundant calculation directly degrades the performance (for example, one 64x64 prediction unit needs up to 256 4x4 pipeline blocks, even if pixels in 64x64 block share one motion vector).

To overcome the redundancy problem, we propose the CU-adaptive pipeline granularity. In detail, pipeline granularity is coding unit. Though the size and partition mode are various for different coding unit, the CU-adaptive pipeline can still have several advantages. Firstly, compared to the previous fixed pipeline blocks, the CU-adaptive pipeline scheme can efficiently eliminate the redundancy for larger CU cases, which is considered to happen frequently in 8K UHDTV. Secondly, instead of PU based pipeline, the CU-adaptive pipeline scheme can reduce the implementation complexity for pipeline controller. The reason is the proposed scheme can reduce the number of possible cases for pipeline granularity, especially for asymmetrical partitions. Thirdly, in spite of various sizes, the process for each CU follows similar procedure, leading to easier implementation. Finally, even if CU contains two or more PU, neighboring blocks for each PUs have overlap, which means that data reuse can be utilized for memory bandwidth reduction, as shown in Fig. 4(a).

A simplified data flow in Fig. 4(b) illustrates that three pipeline processing stages are designed for unified PDec. Considering BS calculation needs the result of current block’s MV, so BS calculation should be scheduled after the correspondent process for MV. Thus, by incorporating the two processes together, we define the unified pipeline as consisting of three main steps [14]: 1) memory reading for reference data fetching, 2) MV calculation for current CU and 3) BS calculation and MV writing back. Notice that only BS calculation for PU edge is in stage 3 while TU edge is done outside pipeline. The reason is that BS calculation for PU edge require MV information so it ties tightly with the rest of CU-adaptive pipeline. Meanwhile, BS for TU edge is quite simple so that it will not increase the hardware complexity much by separating it from PU edge.

Table 1 shows the cycle resource allocation for each stage and for each pipeline granularity. Generally, a pipeline’s throughput is mainly decided by the slowest stage. Hence the performance will not be degraded so much if we decelerate the faster stages. Thus we unify the process speed of each granularity as the slowest stage among all three stages, which is shown in the Max. Cycles column in Table 1. Notice that the data in brackets indicates that a CU contains two PU. It is achieved by adding no operation (NOP) cycles in the trailing of fast stages. The merit is the implementation for pipeline controller is greatly simplified. The throughput for each CU sizes are also listed in the table, which proves that the proposed PDec is competent for at least 8 pixels per cycle.
Fig. 4 Analysis on designing the CU-adaptive pipeline.

Table 1 Cycles allocation for different CU cases.

<table>
<thead>
<tr>
<th>CU Size</th>
<th>Process Cycles for Each Stage</th>
<th>Max. Cycles for each CU</th>
<th>Throughput (pixel/cycle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8x8</td>
<td>(Cycles for CU contains 2 PUs are shown in brackets)</td>
<td>8(8)</td>
<td>8.0(8.0)</td>
</tr>
<tr>
<td>16x16</td>
<td>10(12)</td>
<td>7(10)</td>
<td>25.6(21.3)</td>
</tr>
<tr>
<td>32x32</td>
<td>18(20)</td>
<td>15(22)</td>
<td>56.9(46.5)</td>
</tr>
<tr>
<td>64x64</td>
<td>34(36)</td>
<td>31(46)</td>
<td>120.5(89.0)</td>
</tr>
</tbody>
</table>
3.2 Blockdiagram for the CU-adaptive Pipeline

Fig. 4(d) shows the whole framework of unified parameter decoder. The below part illustrates the data flow inside the proposed CU-adaptive pipeline. Inside each pipeline stage, important process units are depicted. The detail for these three stages will be given in the following Sect. 3.3–3.5, correspondingly. The upper part gives a brief representation for memory in this design, which will be discussed in Sect. 4. Based on the pipeline analysis, no matter what the current and next CU’s partition mode and size are, they can be processed continuously without pipeline pause to guarantee high pipeline performance.

3.3 Pipeline Stage 1

The first stage is in charge of fetching neighboring data from memory, which will be explained in Sect. 4. Generally, each PU in CU needs four cycles for fetching MV reference blocks’ information, as shown in Fig. 4(c). When current CU contains more than one PU, only extra two cycles are needed for second PU as the rest of the neighboring blocks can be reused from first PU. After finishing MV reference fetching, BS reference blocks are accessed and then pushed into FIFO in preparation for stage 3.

Three memory controllers work in parallel as shown in Fig. 4(c). Rather than sequential process, these controllers in parallel is to improve the throughput in worst case. The worst case is defined that CU size is 8x8 and partition mode is symmetric partition (two PU in this CU). For each PU, five spatial neighboring in line buffer and 2 temporal neighboring in collocated SRAM need 7 cycles to be fetched. Extra 3 cycles are for fetching BS neighboring blocks. Thus, totally 20 cycles (ten for each PU) are required for this worst case. Under this situation, the performance can’t even support a 8K@30fps video application. Therefore, we propose three memory controllers working parallel. Together with the data reuse scheme in Fig. 4(a), the design can achieve throughput for 8K@60fps.

3.4 Pipeline Stage 2

The algorithm-irregular MV calculation (AMVP mode and merge mode) is schedule in Stage 2. The implementation suffers from huge area cost and timing issue. Hence, we propose resource sharing scheme and index-mapping scheme for optimization.

Firstly, MV scaling calculator is optimized and reused. The data path of scaling calculator is depicted in Fig. 5. In this procedure, one division, two multiplications and several adders are employed. Without optimization, the synthesis result reports 7ns critical path and around 10k gates cost. The critical path is marked as read in Fig. 5. Such serious costs in both timing and area are unacceptable for an efficient design. Sub-stage pipeline is proposed inside stage 2 for scaling calculator. Four sub-stages are designed in Fig. 5, which are segmented by dotted line. By doing so, a balanced division for critical path is achieved. Meanwhile, LUT replaces the division implementation in the sub-stage 2 for area and timing saving. The proposed sub-stage pipeline is capable for working under 2.5ns timing constrain. As one scaling calculator is capable to process one motion vector, two calculators are necessary for scaling two motion vectors in a bi-directional prediction case. Thus the optimized architecture for scaling calculator can achieve 400 MHz timing constrains.

Secondly, index-mapping scheme for merge mode engine is proposed. Candidate Producer in Fig. 6(a) follows the same procedure and accomplishes the identical function for merge mode in HEVC standard. Taking Combined Candidate Producer as an example, we receive available candidates (OrigCand) from 5 spatial and 2 temporal candidates.
3.5 Pipeline Stage 3

This stage is in charge of calculating BS parameters and writing MV result back into memory. BS calculation relies on information of prediction unit and transform unit, as is shown in Fig. 4(d). Split 4x4 pipeline scheme inside stage 3 is proposed for pipeline buffer reduction. The CU-adaptive pipeline is proposed in the top level of PDec design. However, when CU size is 64x64, totally 32 (64/4*2) neighboring 4x4 blocks should be buffered before the start of BS calculation. We notice that BS calculation is executed 4x4 by 4x4, thus a further separation for pipeline granularity is feasible. By applying split 4x4 pipeline scheme, only 8 neighboring blocks (the delay between stage 2 and 3) are needed to be buffered, which helps reduce 75% of buffering memory.

Process speed is analyzed for split 4x pipeline scheme. In stage 3 only left and top edges of a PU is calculated to get an internal BS, which will be further refined by TU information as shown in Fig. 4(d). For example, if a PU is completely a PU for a 64x64 CU, 64/4=16 4x4 blocks on PU left edge needs 16 cycles while another 16 cycles are for the PU top edge. Considering left and top PU edges have a 4x4 block overlap, totally (16+16-1)=31 cycles are required. 46 cycles are for the case where CU contains two PU so that one more PU edge inside CU will further cost another (16-1)=15 cycles compared to the previous case, as is shown in Table 1.

4. Proposed Memory Organization

4.1 Memory Organization for Spatial Storage

In parameter decoder design, data accessing for spatial neighboring is considered as the bottleneck for achieving high throughput requirement of 8K UHDTV application. As is mentioned in Sect. 2, five neighboring blocks are needed for MV process and all left and top neighboring blocks for BS calculation. All the data should be fetched within the scheduled cycles listed in Table 1. In order to meet the throughput, line buffer is maintained as shown in Fig. 7. Considering the ability to distinguish different blocks, the proposed line buffer’s cell is set equal to minimum TU size (4x4). Meanwhile, to simplify the control logic, single line buffer is employed to consist of not only the storage of the top row (Top-Buffer) but also the left blocks of current CTU (Left-Buffer). Under this memory organization, the neighboring block A0, A1 are stored in Left-Buffer while B0, B1 are in Top-Buffer correspondingly. For each prediction unit, these four blocks can be read out from line buffer in four cycles. If CU is divided into two PUs, at most two blocks (A0, B1 or A1, B0) can be reused so extra two cycles is needed. Further, A1 and B1 can be reused for BS calculation to save extra two cycles.

In addition, fifteen registers are maintained to store the
top-left B2 blocks. One is that the proposed replacing strategy for line buffer can’t store the blocks at the concave corner in the decoded regions. So the top-left registers help implement the deflection of line buffer. On the other hand, we use register instead of reusing line buffer for B2 storage. The reason is that B2 will be read and refreshed frequently inside CTU, implementation by register will eliminate the potential memory accessing conflict problem (read and write simultaneously). The Left-top register consists of fifteen identical registers, each of which is 95 bits. The content of these 95 bits is the same as that in line buffer, which is shown in Fig. 7. Whenever a block at concave corner is written into line buffer, it is also written into correspondent Left-top register simultaneously. Thus there is no communication between line buffer and Left-top register, so that further memory conflict problem is avoided.

The proposed pipeline stages in Sect. 3.1 leads to potential hazards for memory access. Forward by-passing is used to eliminate the problems. In detail, we allocate memory reading and writing in stage 1 and stage 3 respectively. Thus two potential memory conflict hazards exist. Firstly, the same memory address may be read and written in one cycle simultaneously. The second is write-after-read hazard caused by delayed writing operation. For these two hazards, two by-pass registers are inserted between stage 1 and 2 as shown in Fig. 4(d). The proposed pipeline fixes the delays between stages so that stage 3 is always 12 cycles delayed than stage 1. Meanwhile, the minimum process cycles are 8 cycles when CU is 8x8. Therefore, when current CU is on its third stage, at most following two CUs go through its first stage. Potential memory hazards can only happen during these 10 cycles. Thus we use registers to store the previous two CUs’ result. Whenever current CU wants to access contents of previous two CU, we directly fetch them from registers instead of reading from line buffer. Therefore the forward by-pass scheme can efficiently deal with the potential hazards for the line buffer.

4.2 Memory Organization for Temporal Storage

In this section PU-based coding scheme and the cyclically mapped SRAM are introduced in detail. All the co-located data is stored in the off-chip DRAM because of the huge data volume. Generally, DRAM bandwidth issue is regarded as the bottleneck for the real-time video decoder. In terms of 8K UHDTV, more attentions should be paid to relieve the DRAM bandwidth burden.

The PU-based coding scheme is proposed for DRAM bandwidth reduction. Writing data back to DRAM for each 4x4 blocks, just like the way of line buffer, is unacceptable because of the huge data volume. We notice that only the top-left corner 4x4 block in each 16x16 block will be used as the temporal candidate in HEVC standard, which means we only need to store one set of prediction parameters for each 16x16 block for col-located storage. It will help reduce 93.75% of DRAM bandwidth. Further, considering the prediction unit larger than 16x16, the same co-located information is repeatedly written into DRAM for each 16x16, leading to meaningless burden on bandwidth requirement. Thus we propose the PU-based coding scheme for co-located data compression. The data stored in DRAM contains not only the prediction parameters but also the 8-bit description for current PU in CTU. As depicted in Fig. 8, a 32x32 PU will write identical data (95 bits) into DRAM for four times. PU-based scheme will avoid such kind of redundancy by writing this identical data (PU info.) only once with 8 trailing bits. Though extra 8 bits for PU description is added for each co-located block, the redundant DRAM storage is avoid. The experiment result shows that by applying the PU-based coding scheme, further 30% of bandwidth requirement reduction can be achieved.

Figure 9 illustrates the proposed cyclic SRAM. It is designed to support the random access characteristic for accessing co-located data. As is mentioned above, the co-located data is pre-coded before storage. In the reading side decoding process is first done. After that the decoded data will be written into the SRAM for random access. The pro-
posited cyclic SRAM can be regarded as an addressable FIFO whose size is four CTU’s co-located information. Based on the analysis on HEVC, the current decoding block will only use the co-located CTU and the col-located next CTU’s data for MV calculation, as shown in Fig. 9. Thus, as long as col-located following two CTUs’ content is stored inside, the cyclic SRAM can be read by PDec. In the non-read cycle, cyclic SRAM will keep writing decoded data until the cyclic SRAM meets the full condition, when writing cursor is equal to the write stop address. Single-port SRAM is capable of above functions so as to reduce control logic and save area cost.

### 5. Implementation Result

#### 5.1 Simulation on PU-Based Coding Scheme

PU-based coding scheme helps to reduce the DRAM bandwidth (BW). As this work is aiming at UHDTV application, we choose ClassA sequences (Traffic and PeopleOnStreet) which is 2560x1600 with 300 frames and two 7680x4320 video sequences (06000 and 12000) with 150 frames as our test sequences. Different configurations (RandomAccess (RA) and Lowdelay (LD)) and QP value are considered. The result is shown in Table 3.

In Table 3 all the bandwidth is normalized to frame width is twice larger than 4K’s, leads to double size of storage for 4x4 block granularity is needed; 2) MV parameter in HEVC is more than that in H.264; 3) 8K UHDTV’s frame width is twice larger than 4K’s, leads to double size of line buffer. Finally, as total area is related to the throughput, in Sect. 4.2, necessary DRAM BW is 92.3MB/s. PU-based coding scheme can further reduce the bandwidth by at least around 30%. For a 7680x4320 sequence 12000 at QP=40, up to 88.3% BW reduction can be achieved.

#### 5.2 Synthesis Result

The unified PDec is implemented on RTL level design using Verilog HDL and further synthesized by Synopsys Design Compiler with TSMC 90 nm process.

In detail, the maximum clock frequency of our proposal can achieve 400 MHz. The area under 300 HMz is 93.3k. The on-chip line buffer’s size is 23.0 kB. Here we analysis HEVC standard and define the worst is that the whole frame is coded as 8x8 coding unit. From Table 1 we know that the cycles for worst case are 8, equally to 8 pixels per cycle. This throughput is enough for real-time decoding for 7680x4320@60fps video sequence. We also simulate our proposal on HEVC test sequence and the result shows the average process speed is 17.8 pixel/cycle, which is able to finish decoding highest profile 6.3 7680x4320@60fps applications at only around 111.8 MHz on average.

#### Table 2

Comparison with state-of-the-art works.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>MV</td>
<td>MV/BS/Intra</td>
<td>MV/BS/Intra</td>
<td>MV/BS</td>
</tr>
<tr>
<td>Worst-case Throughput (Pixel/cycle)</td>
<td>0.98</td>
<td>0.73</td>
<td>4.0</td>
<td>8.0</td>
</tr>
<tr>
<td>Avg. Throughput (Pixel/cycle)</td>
<td>1.6</td>
<td>1.6</td>
<td>4.0</td>
<td>17.8</td>
</tr>
<tr>
<td>SRAM Size (Line buffer)</td>
<td>2.8k(1080p)</td>
<td>4.75k(1080p)</td>
<td>3.6k(2160p)</td>
<td>23.0k(4320p)</td>
</tr>
<tr>
<td>Logic Gate</td>
<td>52k</td>
<td>63.0k</td>
<td>37.2k</td>
<td>93.3k</td>
</tr>
<tr>
<td>Norm. Logic Gate*</td>
<td>41.80</td>
<td>101.27</td>
<td>7.47</td>
<td>4.69</td>
</tr>
<tr>
<td>Max. Resolution</td>
<td>1920x1080@60fps</td>
<td>1920x1080@30fps</td>
<td>3840x2160@60fps</td>
<td>7680x4320@60fps</td>
</tr>
<tr>
<td>Technology</td>
<td>90 nm</td>
<td>65 nm</td>
<td>90 nm</td>
<td>90 nm</td>
</tr>
</tbody>
</table>

**As the supported maximum resolution varies, we define normalized logic gate equals to logic gate divided by processed pixels per second (x10-5).**
we define the normalized gate number in the table for a fair comparison. The normalized results show that our proposed unified architecture has around 36% efficiency on area cost, even if supported HEVC’s complexity is more than that of H.264/AVC.

Totally, our proposed unified motion vector and boundary strength PDec can support real-time decoding for 7680x4320@60fps video under 249 MHz in worst case.

6. Conclusion

This paper proposes unified parameter decoder architecture for UHDTV. The design can accomplish the algorithm of MV and BS calculation for sharing the memory and logic resources. In particular, CU-based pipeline strategy is proposed to simplify control logic as well as supporting HEVC’s new coding tools. Moreover, on-chip line buffer and cyclic SRAM are designed for both spatial and temporal reference storage to guarantee enough bandwidth requirements. PU-based coding scheme can help reduce around 30% of DRAM bandwidth. Finally, optimization on irregular algorithm is adopted for 43.2k logic gates reduction. In total, the proposed unified parameter decoder supports real-time video decoding for 7680x4320@60fps application at 249 MHz in worst case.

Acknowledgments

This work is supported by Waseda University Graduate Program for Embodiment Informatics (FY2013-FY2019).

References


Appendix: Glossary of Acronyms

Below gives a glossary of acronyms used in this paper in alphabetic order.

<table>
<thead>
<tr>
<th>Acronyms</th>
<th>The full term</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMVP</td>
<td>Advanced Motion Vector Prediction</td>
</tr>
<tr>
<td>BS</td>
<td>Boundary Strength</td>
</tr>
<tr>
<td>BW</td>
<td>BandWidth</td>
</tr>
<tr>
<td>CTU</td>
<td>Coding Tree Unit</td>
</tr>
<tr>
<td>CU</td>
<td>Coding Unit</td>
</tr>
<tr>
<td>DBF</td>
<td>DeBlocking Filter</td>
</tr>
<tr>
<td>MC</td>
<td>Motion Compensation</td>
</tr>
<tr>
<td>MV</td>
<td>Motion Vector</td>
</tr>
<tr>
<td>NOP</td>
<td>No Operation</td>
</tr>
<tr>
<td>PDec</td>
<td>Parameter Decoder</td>
</tr>
<tr>
<td>PU</td>
<td>Prediction Unit</td>
</tr>
<tr>
<td>TU</td>
<td>Transform Unit</td>
</tr>
</tbody>
</table>

Shihao Wang received the B.E. degree in electronic engineering from Shanghai Jiao Tong University, China, in 2012 and the M.E. degree in engineering from Waseda University, Japan, in 2014. He is currently pursuing the Ph.D. degree in engineering in Waseda University, Japan. His interests are in VLSI architectures and algorithms for multimedia signal processing. He was a recipient of Ting Hsin Scholarship in 2013 and 2014. He is being supported by the Graduate Program of Embodiment Informatics held by Waseda University and MEXT, Japan.
Dajiang Zhou received B.E. and M.E. degrees from Shanghai Jiao Tong University, China. He earned his Ph.D. degree in engineering from Waseda University, Japan, in 2010. Dr. Zhou is currently with Waseda as an Assistant Professor at the Graduate School of Information, Production and Systems. His interests are in algorithms and implementations for multimedia and communications signal processing, especially in low-power high-performance VLSI architectures for video codecs including H.265/HEVC and H.264/AVC. Dr. Zhou received a number of awards including the Best Student Paper Award of VLSI Circuits Symposium 2010, the International Low Power Design Contest Award of ACM ISLPED 2010, the 2013 Kenjiro Takayanagi Young Researcher Award, and the Chinese Government Award for Excellent Students Abroad of 2010. Dr. Zhou was a recipient of the research fellowship of the Japan Society for the Promotion of Science during 2009–2011. His work on an 8K UHDTV video decoder VLSI chip was granted the 2012 Semiconductor of the Year Award of Japan.

Jianbin Zhou received the B.E. degree in software engineering from South China University of Technology, China, in 2012 and the M.E. degree in engineering from Waseda University, Japan, in 2014. He is currently pursuing the Ph.D. degree in engineering from Waseda University, Japan. His interests are in VLSI architectures and algorithms for multimedia signal processing. He was a recipient of Ting Hsin Scholarship in 2013 and 2014. He is being supported by the Graduate Program of Embodiment Informatics held by Waseda University and MEXT, Japan.

Takeshi Yoshimura received B.E., M.E., and Dr.Eng. degrees from Osaka University, Osaka, Japan, in 1972, 1974, and 1997. He joined the NEC Corporation, Kawasaki, Japan, in 1974, where he has been engaged in research and development efforts devoted to computer application systems for communication network design, hydraulic network design, and VLSI CAD. From 1979 to 1980 he was on leave at the Electronics Research Laboratory, University of California, Berkeley, where he worked on very large scale integration computer-aided design layout. He received Best Paper Awards from the Institute of Electronics, Information and Communication Engineers of Japan (IEICE) and the IEEE CAS Society. Dr. Yoshimura is a professor of Waseda University and a member of IPSJ (the Information Processing Society of Japan), and IEEE.

Satoshi Goto received the B.E. and the M.E. Degrees in Electronics and Communication Engineering from Waseda University in 1968 and 1970 respectively. He also received the Dr. of Engineering from the same University in 1978. He joined NEC Laboratories in 1970 where he worked for LSI design, Multimedia system and Software as GM and Vice President. Since 2002, he has been Professor, at Graduate school of Information, Production and Systems of Waseda University at Kitakyushu. He served as GC of ICCAD, ASPDAC, VLSI-SOC, ASICON and ISOCC and was a board member of IEEE CAS society. He is IEEE Life Fellow and IEICE Fellow. He is Visiting Professor at Shanghai Jiao Tong University and Tsinghua University of China and Member of Science Council of Japan.