A High-Throughput HEVC Deblocking Filter VLSI Architecture for 8kx4k Application

Wei Cheng, Yibo Fan, YanHeng Lu, Yize Jin, Xiaoyang Zeng
State Key Laboratory of ASIC and System
Fudan University, Shanghai 200433, China
{13212020005, 14210720072, fanyibo}@fudan.edu.cn

Abstract—As the next generation of video coding standard, High Efficiency Video Coding (HEVC) aims to reduce 50% bit rates in comparison with previous video coding standards. In order to increase the Deblocking Filter (DBF) throughput, we propose a memory of ping-pong and interlacing VLSI architecture to prevent DBF from unnecessarily waiting for pixels in both vertical and horizontal, which only takes 435 cycles at worst to process a LCU of 64x64 pixels size. Based on the memory organization, a four stage pipeline with a PreFilter was proposed to eliminate the data dependence in the filter processing and makes it working on 318M possible. As a result, our design proposed to eliminate the data dependence in the filter processing memory organization, a four stage pipeline with a PreFilter was proposed to eliminate the data dependence in the filter processing and makes it working on 318M possible. As a result, our design can support 8kx4k@90fps real-time applications with SMIC 0.13μm technology at the cost of 62.9K gates.

Keywords—DBF; 8kx4k; HEVC; 90fps.

I. INTRODUCTION

High Efficiency Video Coding (HEVC), the latest video coding standard, was developed and published in Jan-2013 by Joint Collaborative Team on Video Coding (JCTVC) [1], which is formed by ISO/IEC Moving Picture Experts Group and ITU-T Video Coding Experts Group. Compared to the previous video coding standard H.264/AVC, on one hand, HEVC promises to reduce 50% bit rates under the same subjective and objective quality; on the other hand, both H.264/AVC and HEVC employ similar block based hybrid approach and transform coding framework.

Unlike H.264/AVC splits a picture into fixed size macroblocks of 16x16 pixels, HEVC divides a picture into coding tree units (CTU) [2]. The CTU can be further split into coding unit (CU) of 8x8, 16x16, 32x32, 64x64 pixels. The largest CU (LCU) is the basic processing unit and can usually be set to 64x64 or 32x32 pixels. In prediction and transform sub parts of HEVC, CUs are divided into prediction unit (PU) and transform unit (TU) respectively. The size of the former supports symmetric and asymmetric sizes and the latter can vary from 4x4 to 32x32 pixels.

This block-based prediction and transform coding method in HEVC and H.264/AVC, usually lead to discontinuities at block boundaries which will significantly increase bit rate. In order to dispel discontinuities, both H.264/AVC and HEVC employs Deblocking Filter (DBF). However, the former is much more complex and takes about one-third of the computational complexity of decoder [4]. On the contrary, the latter is less complex, and will leads to an average bit rate reduction of 1.3-3.3% at the same quality. For certain sequences, more than 6% bit reduction is achieved [1].

In the past few years, there are some literatures on the topic of DBF. In [5], two parallel data paths with 10 SRAMs were applied to increase its performance with the cost of larger areas and memory management complexity, meanwhile its performance is not very high which can only support 1920x1080@30fps real time encoding. In [6], an architecture with a novel memory interlacing organization and four-stage pipeline is proposed, however it only processes 32x32 LCU and the pixels loading and output are not considered. In [7], an architecture combining Deblocking and SAO which can reach 266 MHz at most in 28 nm CMOS technology is proposed, however the throughput is 8kx4k@20fps which can’t support 8kx4k real time encoding. In fact, the filter procedure optimization is not considered or mentioned in [5], [6] and [7].

In this paper, we present a high throughput architecture with novel memory organization and filter order to reduce processing cycles and on-chip memory size. At the same time a four stage pipeline with a PreFilter to eliminate the data dependence and reduce the critical path is employed to increase the working frequency and throughput. The rest of this paper is organized as follows: Section II briefly introduces the HEVC DBF algorithm. Section III presents the hardware architecture in detail and in section IV implementation result is shown. Finally section V concludes this paper briefly.

II. DEBLOCKING FILTER ALGORITHM

In HEVC, DBF is applied to the boundaries of the 8x8 blocks. In fact, there are two kinds of boundaries in HEVC: vertical boundaries and horizontal boundaries. In the reference software HM, vertical boundaries are filtered firstly; and after all vertical boundaries are filtered, horizontal boundaries are filtered.

Every boundary is assigned a Boundary Strength (BS) value depending on the coding modes and conditions of current CU, such as prediction mode, motion vector (MV) and so on. Its value can be 0, 1 or 2. For chroma filtering is done only when the BS value equals to 2, while for luma as long as the BS value is not 0.

The filtering process of each boundaries can be divided into two sub-processes as shown in Fig. 1: 1) making a decision (MD) on whether a boundary is filtered or not; 2) how a boundary is filtered (HBF) if it has to be filtered. For the first sub-process, the following data are needed: 1) Boundary...
Strength (BS) to indicate the filter strength; 2) Quantization Parameter (QP) determining β and Tc threshold values; 3) P block and Q block: two adjacent 4x4 block pixels to be filtered. Based on the two adjacent 4x4 block pixels, two parameters d and de are calculated, then Tc, β, d and de will decide how a boundary is filtered: 1) strong filtering: three pixels are modified; 2) weak filtering: one or two pixels are modified.

In order to access the data smoothly and reduce the I/O bandwidth between the chip and system, we adopt two layers mapping and on-chip memory to store the 4x4 blocks.

### TABLE I. MEMORY SUMMARY

<table>
<thead>
<tr>
<th>Heading</th>
<th>Size</th>
<th>Memory Structure</th>
<th>LODING/cycles</th>
<th>FILTERING/cycles</th>
<th>OUTPUT/cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM0</td>
<td>68x128</td>
<td>LLUMA, CB, CR</td>
<td>68</td>
<td>132</td>
<td>68</td>
</tr>
<tr>
<td>SRAM1</td>
<td>68x128</td>
<td>LLUMA, CB, CR</td>
<td>68</td>
<td>132</td>
<td>68</td>
</tr>
<tr>
<td>SRAM2</td>
<td>68x128</td>
<td>LLUMA, CB, CR</td>
<td>68</td>
<td>132</td>
<td>68</td>
</tr>
<tr>
<td>SRAM3</td>
<td>68x128</td>
<td>LLUMA, CB, CR</td>
<td>68</td>
<td>132</td>
<td>68</td>
</tr>
<tr>
<td>SRAM4</td>
<td>72x128</td>
<td>LLUMA, CB, CR</td>
<td>72</td>
<td>140</td>
<td>72</td>
</tr>
<tr>
<td>SRAM5</td>
<td>72x128</td>
<td>LLUMA, CB, CR</td>
<td>72</td>
<td>140</td>
<td>72</td>
</tr>
</tbody>
</table>

1) Logic Mapping: A 64x64 LCU consists of 256 4x4 luma blocks, 64 4x4 chroma cb blocks and 64 4x4 chroma cr blocks. Since the numbers of 4x4 blocks differ from each other quite extensively, we regroup these 4x4 blocks as well as the 4x4 block of the left LCU and the top LCU into 3 parts as shown in Fig. 3. Each square in Fig. 3 stands for a 4x4 block. The three parts are described as following:

- **a) LLUMA**: consisting of the 4x4 blocks whose prefix is L or R. The total number of 4x4 blocks is 136.

Fig 1. Deblocking filter flow

Fig 2. Proposed DBF hardware Top architecture
b) **RLUMA**: consisting of the 4x4 blocks whose prefix is X or Y. The number of 4x4 blocks is 136.

c) **CCBC**: combination of Chroma cb and Chroma Cr, and the prefix of the 4x4 block is C or D. The number of 4x4 blocks is 144.

2) **Physical Mapping**: After the Logic Mapping, the numbers of the 4x4 block of three parts are almost the same. Then in physical mapping, each part is stored in 2 two ports SRAMs working in a memory interlacing style as shown in Fig. 3 and Fig. 4 and summarized in Table I. Thus Two 4x4 blocks on both sides of each boundary always come from different SRAMs. As a result, all pixels can be easily accessed from the SRAMs for both vertical and horizontal filtering operations.

C. **Data Flow**

The 6 two ports SRAMs working in a ping-pong style and their working state can be divided into three states according to relationship between the 6 SRAMs and external memory and the ProcessingEngine.

1) **LOADING**: Each cycle loads a 4x4 block from external memory, thus SRAM0, SRAM1, SRAM2 and SRAM3 take 68 cycles respectively, while SRAM4 and SRAM5 take 72 cycles respectively for CCBC has two top LCUs, One is for Chroma Cb and the other is for Chroma Cr.

2) **FILTERING**: Perform filter operation on each boundary in a pipelined manner. All LLUMA, RLUMA and CCBC have 128 edges to filter and spend 132 cycles to filter.

3) **OUTPUT**: Output 4x4 blocks to external memory after all boundaries have been filtered. The cycles spent are the same with LOADING.

D. **SwitchingMatrix**

At different filter stage, a 4x4 block from the same SRAM acts as different roles in the ProcessingEngine because DBF is designed to perform both vertical and horizontal filtering. For example, the top right 4x4 block is Q block when critical edge is filtered, while P block when horizontal edge is filtered done. The aim of this module is to send the two adjacent 4x4 blocks to P block and Q block and write back to the SRAMs after filtered correctly.
E. Processing Engine

The Processing Engine is the critical module of the system limiting the maximum working frequency due to the filter operation, especially normal filter as shown in Pseudocode 1. In order to reduce the latency, all the filtering operations are done in one cycle. However, the variables $\triangle p$ and $\triangle q$ are depending on $\triangle$ which will expend the critical path. Thus, we use a PreFilter to calculate $\triangle$ one cycle ahead so as to eliminate the data dependence and break the critical path.

Based on the PreFilter and SwitchingMatrix, a four stage pipeline architecture of the filtering process is proposed. The result of DC synthesis shows that the filter operation is not the critical path any longer in this kind of architecture. Each stage of the pipeline is described as following:

Stage 1: Reading pixels from the memory and splitting them to the input signals using the SwitchingMatrix, the BS and QP order to reduce the latency, all the filtering operations are done one cycle ahead so as to eliminate the data dependence and break the critical path.

Stage 2: Some threshold values are calculated to determine how the pixels to be filtered. Meanwhile this stage contains PreFilter module to calculate $\triangle$ used for normal filter.

Stage 3: The pixels were filtered by strong filter and normal filter, and the appropriate one is selected as the correct filter according the threshold values calculated in the previous period and the BS.

Stage 4: The SwitchingMatrix is used to combine all the 8-bit pixels of the same 4x4 block into a 128 bits signal and then write it back to the memory.

IV. IMPLEMENTATION RESULTS

The proposed DBF VLSI architecture is written in verilog and synthesized in 0.13um technology. The comparison among some previous work and the proposed design is drawn in Table II. As Table II shows, our design can achieve 318MHz working frequency at most, which makes it capable of supporting 8kx4k@90fps and is much higher than [5] and [7]. The throughput in [6] is close to ours, however it does not consider the cycles of data loading and output. If it considers these cycles, its true throughput is about one third of 56 fps.

V. SUMMARY AND CONCLUSION

In this paper, a DBF VLSI architecture is proposed which splits the luma component into two parts: LLUMA and RLUMA, combines the chroma cb and cr together, and then filter them in a unique order. In the filter process, a PreFilter is used to eliminate the data dependence. The synthesis result shows that the proposed architecture has processing cycles and higher working frequency and throughput making 8kx4k@90fps possible.

ACKNOWLEDGEMENTS

This paper is supported by National Natural Science Foundation of China (61306023), Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP, 2012071120021), STCSM (13511503400), National High Technology Research and Development Program (863, 2012AA012001).

REFERENCES


