ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.10, No.7, 1–9

Register array-based VLSI architecture of H.265/HEVC loop filter

Jongwoo Bae[a)]

[1] *Department of Information and Communication Engineering, Myongji University, San 38–2 Namdong, Cheoin-Gu, Yongin-Si, Gyeonggi-Do 449–728, Korea*

a) *jwbae@mju.ac.kr*

Abstract: A high-performance VLSI architecture for H.265/HEVC loop filter is proposed. The architecture is implemented by a parallel register array that consists of 8 × 8 registers with two data flow directions. The register array computes the sub-blocks of four different coding tree blocks at the same time based on the analysis of the computation order. This leads to the small number of registers used in the register array. The architecture computes 4 K UHD at 30 fps in real-time. The size of the synthesized design is 54 K gates. The operating clock frequency is 225 MHz in TSMC 65 nm process. If the degree of parallelism is increased to four, the architecture can compute up to 8 K UHD at 60 fps.

Keywords: H.265, HEVC, Loop Filter, VLSI, register array

Classification: Integrated circuits

References

[1] JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, "High efficiency video coding (HEVC) text specification draft7," May 2012.

[2] J. Bae, N. Park, and S. Lee, "Register Array Structure for Effective Edge Filtering Operation of Deblocking Filter," *LNCS*, vol. 4096, pp. 805–813, Aug. 2006.

[3] T. Tsai and Y. Pan, "High Efficient H.264/AVC Deblocking Filter Architecture for Real-time QFHD," *IEEE Trans. Consum. Electron.*, vol. 55, no. 4, pp. 2248–2256, Nov. 2009.

[4] C. Chen and C. Chen, "Configurable VLSI Architecture for Deblocking Filter in H.264/AVC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 8, pp. 1072–1082, Aug. 2008.

[5] J. Bae and J. Cho, "A decoupled architecture for multi-format video decoder," *IEICE Electron. Express*, vol. 5, no. 18, pp. 705–710, Sept. 2008.

[6] J. Bae, J. Cho, B. Kim, and J. Baek, "High Performance VLSI design of run_before for H.264/AVC CAVLD," *IEICE Electron. Express*, vol. 8, no. 12, pp. 950–955, June 2011.

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

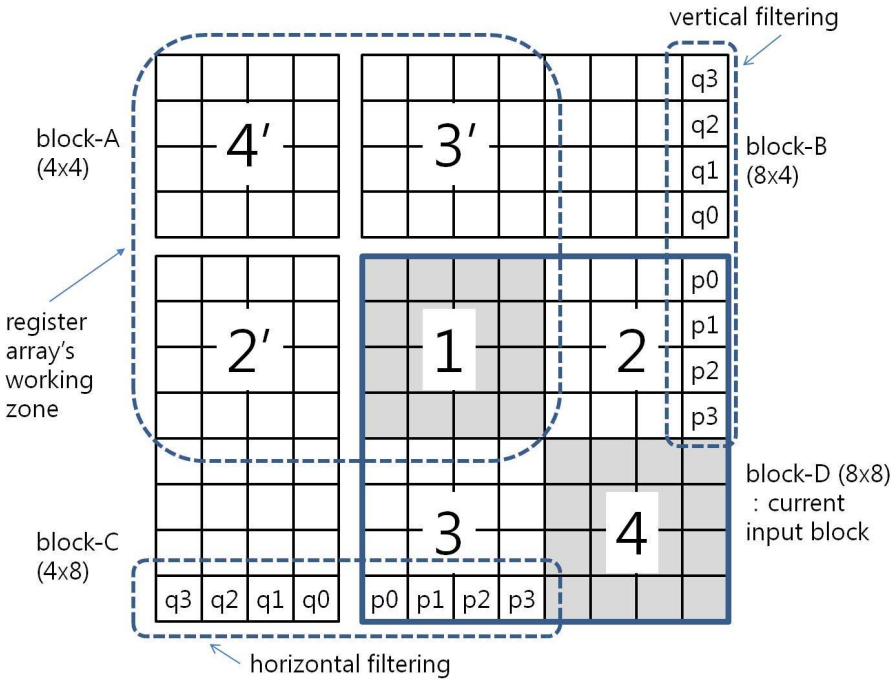

Fig. 1. H.265/HEVC Loop Filter operation

1 Introduction

Recently, a new video codec standard called H.265/HEVC is introduced [1]. It shows the compression performance twice as high as that of H.264/AVC. H.265/HEVC will be adopted for many video applications such as D-TV, Blu-ray, and mobile devices in the near future. In this paper, we propose a novel VLSI architecture for the design of the loop filter of H.265/HEVC, which handles video applications up to 8 K UHD at 60 fps.

A video codec consists of various computational modules such as entropy codec, DCT transform, quantization, motion estimation, prediction, and loop filter. During the compression process, an image frame is divided into smaller blocks called *macroblocks* that are the basic unit of compression. Due to the individual processing of each macroblock, the restored image has block artifacts which indicates the distortion of data discontinuity at the macroblock boundary. A loop filter (also called a deblocking filter) is a computation process to reduce the block artifacts by smoothing the pixels around the block boundaries. The loop filter is one of the most computationally demanding modules of a video codec.

In H.265/HEVC, a frame is divided into coding tree units of 64×64 pixels. The coding tree units are divided again into smaller blocks of 4×4, 16×16, 32×32, and 64×64. The main difference between the loop filters of H.264/AVC and H.265/HEVC is the size of block on which the loop filter is applied. For H.264/AVC, the block size of 4×4 is used. For H.265/HEVC, the block size of 8×8 is used. Fig. 1 shows the example of horizontal and vertical filtering of H.265/HEVC. The horizontal filtering is performed on the left and right boundary edges of the coding tree block. The vertical filtering is performed on the top and bottom boundary edges of the coding

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

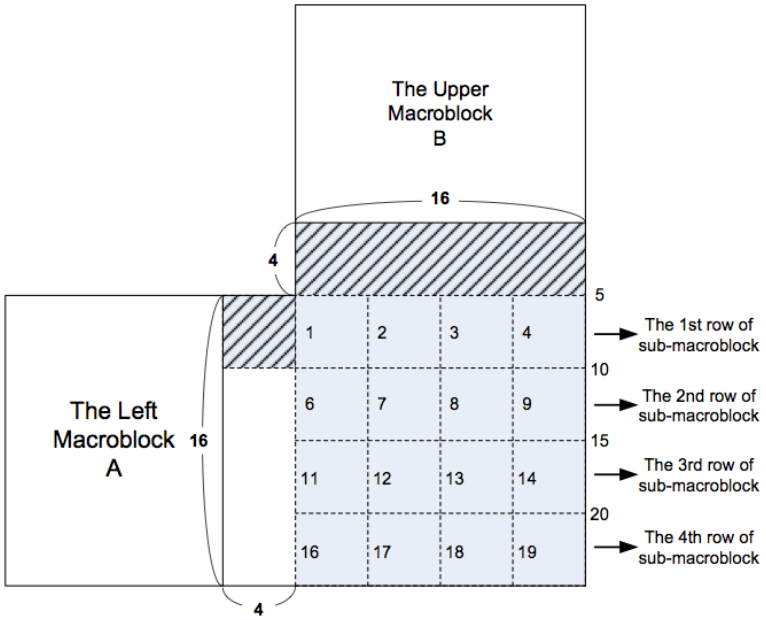

Fig. 2. H.264/AVC loop filter operation

tree block. There are eight filtering positions on each side of a coding tree block. Note that the length of the filter used is still the same as that of H.264/AVC (p0∼p3, q0∼q3). The 8×8 coding tree block is divided into four 4×4 blocks denoted as 1, 2, 3, and 4 for the convenience of explanation in Fig. 1.

Various VLSI architectures have been researched for the previous video codecs including H.264/AVC [2, 3, 4, 5, 6]. For the design of loop filter or deblocking filter for H.264/AVC, many works have been done too [2, 3, 4]. However, there is no known work for the VLSI design of the loop filter for H.265/HEVC yet. Bae [2] introduced a register array-based architecture for H.264/AVC deblocking filter. It is a high-performance parallel architecture, but it supports H.264/AVC only. The register arrays used can be further reduced in the proposed architecture. Tsai [3] and Chen [4] also showed high-performance VLSI architectures with registers and transpose memories for the loop filter of H.264/AVC. However, the architectures cannot be used for H.265/HEVC, since the changed syntax is very different, and the target video size is increased drastically.

This paper is organized as follows. Section 2 shows the related works of loop filter design. Section 3 proposes VLSI architecture for loop filter of H.265/HEVC in detail. Section 4 shows the performance analysis of the proposed architecture. Finally, section 5 concludes the paper.

2 Related works

The concept of register array-based loop filter design is first introduced in [2]. Fig. 2 shows the loop filter problem of H.264/AVC. The loop filter computation for H.264/AVC is performed on the 4×4 block boundaries. Each 4×4 block requires the computation of two horizontal filtering and two vertical filtering at the surrounding four edges of the block. A register

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

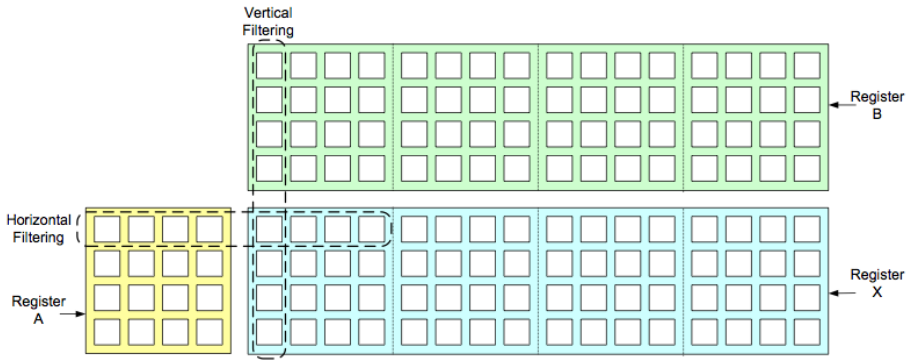

Fig. 3. Register array architecture for H.264/AVC loop filter

array-based architecture shown in Fig. 3 is used to store the data for the computation of 16×4 sub-block of a 16×16 macroblock in [2]. Each 4×4 block is dependent on each other. They cannot be computed in parallel. Therefore, the horizontal filtering is performed on the leftmost 4×4 block first. Then the data is shifted horizontally in a cyclic fashion until all the horizontal filter computations are done for the four 4×4 blocks. After the horizontal computation is over, the vertical filtering can start on the top edge of the leftmost 4×4 block. After one more cyclic shifting of the entire 8×4 block, the computation of loop filter for the 8×4 block is almost finished. It still requires the vertical filtering on the bottom edge. Therefore, the data is shifted upward into register B from register X, and new 8×4 block of the macroblock is loaded into register X. The detail explanation of this architecture can be found in [2].

The architecture works very well for H.264/AVC loop filter, which requires the filter computation on all the four sides of a 4×4 block. For H.265/HEVC, the basic block for loop filter computation is an 8×8 coding tree block. It consists of four 4×4 blocks of 1, 2, 3, and 4 as shown in Fig. 1. These blocks are independent of each other for filter operation, which is the main difference between H.264/AVC and H.265/HEVC. Each 4×4 block has only two sides used for the horizontal or vertical filter operation. Therefore, the 20×8 register array architecture in [2] works slowly for H.265/HEVC due to unnecessary data movement. In addition, the architecture does not fit into the loop filter computation for H.265/HEVC, which requires the computation of the 8×8 block consisting of 4×4 blocks 4', 3', 2', and 1 as shown in Fig. 1. In the next section, we show new register array architecture for H.265/HEVC, which uses smaller register array of 8×8 instead of 20×8.

3 Proposed architecture

The I/O data bandwidth of H.265/HEVC loop filter for 4 K UHD at 30 fps is more than 750 MB/sec. To process 8 K UHD at 60 fps, the required data bandwidth jumps up to 6 GB/sec. Without a highly parallel VLSI architecture, we cannot compute these problems in real-time. We propose a high-performance register array-based VLSI architecture to meet the real-time

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

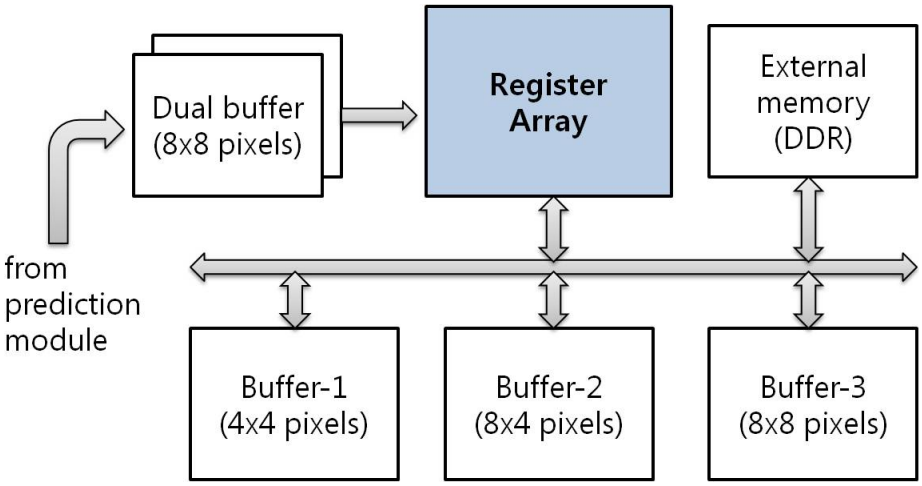

Fig. 4. The block diagram of the proposed H.265/HEVC Loop Filter

requirement of these applications by moving and computing the large data in parallel.

The block diagram of the proposed loop filter is shown in Fig. 4. An 8×8 coding tree block is transferred into the dual input buffer of the loop filter from the previous computation module (*Prediction*). The input coding tree block corresponds to the block D (8×8) of Fig. 1. We also need to input adjacent blocks A (4×4), B (8×4), and C (4×8) for the computation of the current block D. The block named *register array* in Fig. 4 is the main computation block proposed in this paper.

Fig. 5 shows the architecture of the proposed register array. It consists of the four register array blocks of 4×4 named A, B, C and D. They have two directions of data movement, the one from bottom to top, and the other from right to left. H1∼H4 and V1∼V4 are the filters to perform the horizontal and vertical loop filter operations. Depending on the degree of parallelism required, we can select the number of filters from 2, 4, 6, and 8. In Fig. 2, we show the register connections when one horizontal filter (H1) and one vertical filter (V1) are used. It is easy to increase the parallelism to achieve higher performance by changing the register connections and adding more filters.

To understand the operation of the register array, we need to know the computation order of the blocks 1, 2, 3, and 4 in an 8×8 coding tree block shown in Fig. 1. When the current 8×8 coding tree block is input into the loop filter module, we cannot finish the computation of the entire 8×8 block yet. We can only compute the block 1 at the time of input. Block 2 is stored in the on-chip SRAM (buffer 1) to be computed when the next coding tree block is input. It will be computed as block 2' for the computation of the next coding tree block as shown in the register array's working zone in Fig. 1. Blocks 3 and 4 are stored in the external memory first. Then, they are read into the register array as blocks 3' and 4' later. Block 3' is computed when the coding tree block below the current coding tree block is computed. Block 4' is computed with the coding tree block next to this.

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

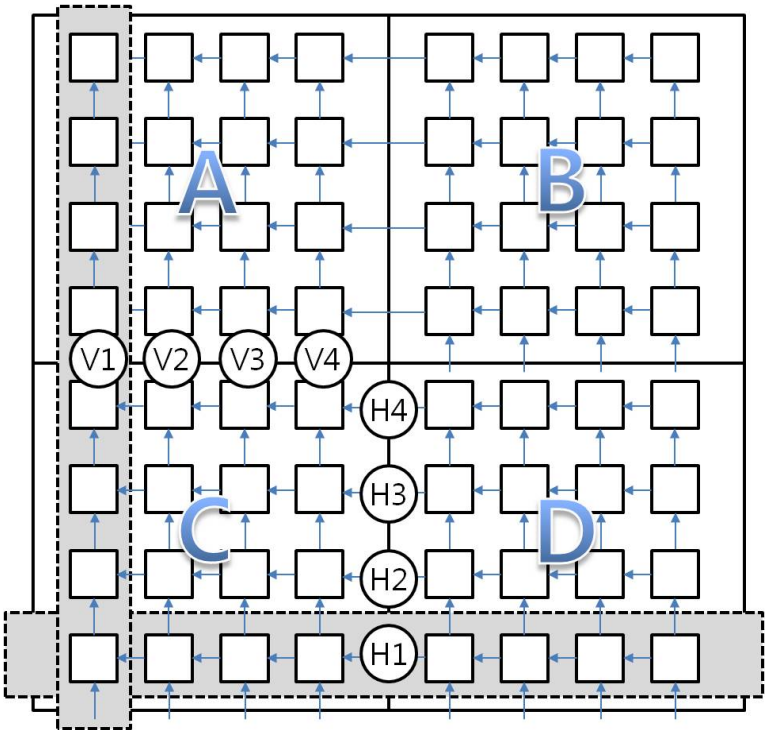

Fig. 5. Register Array Architecture for H.265/HEVC Loop Filter

The data is fed into the register row at the bottom in Fig. 5. The data for block D comes from the block 1 of the current coding tree block stored in the dual buffer. The data for block C comes from an on-chip SRAM (buffer 1) which is the block 2 of the previous coding block (noted as 2' in Fig. 1). The data for block A and B comes from external memory which is stored in the on-chip SRAM (buffer 2). The block A is used for the block 4 of the upper left coding block (noted as 4' in Fig. 1). The block B is used for the block 3 of the upper coding tree block (noted as 3' in Fig. 1).

The filter H1 performs the horizontal filter operation for one clock cycle. After the computation is over, the result is stored in the second register row from the bottom. At the same time, the next input is fed into the register row at the bottom. The same pattern of computation is repeated for 8 clock cycles and the register arrays are filled with horizontally filtered data. The data for block A and B (noted as 4' and 3' in Fig. 1) are fed in first followed by the data for block C and D (noted as 2' and 1 in Fig. 1).

Then, the register arrays shift to the left, and the vertical filter operation is performed at the leftmost column of the register array. The vertical filtering of the register array takes another 8 clock cycles. Therefore, the computation time taken for the current coding tree block is 16 clock cycles in total. The computed data for A, B, C and D is stored in the output buffer (buffer 3), and sent to the external memory. The external memory access can be overlapped while the register array operation is performed in parallel.

An example is presented for easier understanding of the register array operation explained above. Refer to the picture of the coding tree blocks

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

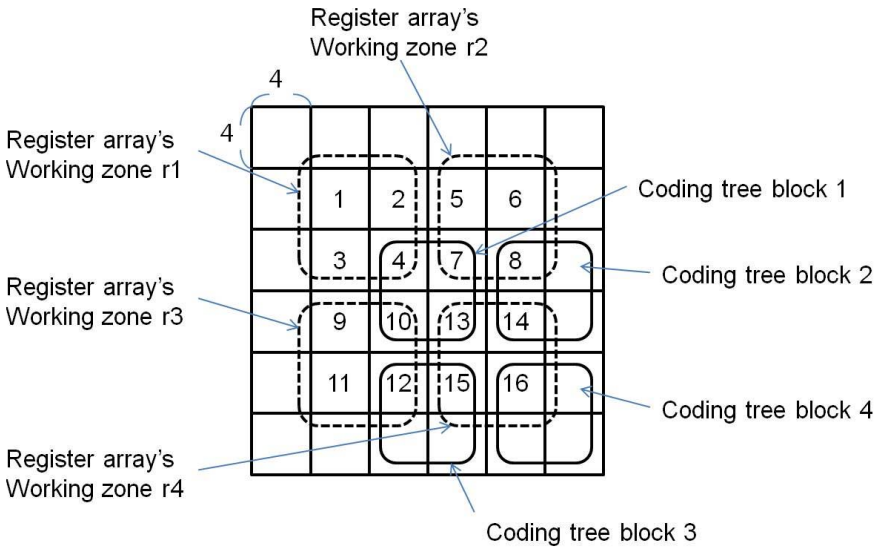

Fig. 6. The computation flow of H.265/HEVC Loop Filter

and the register array's working zones in Fig. 6. The register array's working zone is indicated as a dotted box in the example. The coding tree block 1 consists of four blocks named 4, 7, 10, and 13 in Fig. 6. Observe how each block of the coding tree block 1 is computed in the following explanation.

Step 1. Block 4

For the computation of the block 4 of the coding tree block 1, the register array's working zone is located in the position r1 as shown in Fig. 6. At this position, it loads the data of block 1, 2, 3, and 4 into the register array's block A, B, C, and D respectively. Then the filter computations for the blocks 1, 2, 3, and 4 are performed. For the coding tree block 1, the computation of block 4 is finished. The data for block 3 comes from buffer 1, and the data for block 1 and 2 comes from buffer 2.

Step 2. Block 7

For the computation of the block 7 of the coding tree block 1, the register array's working zone is located in the position r2 as shown in Fig. 6. At this position, it loads the data of block 5, 6, 7, and 8 into the register array's block A, B, C, and D respectively. Then the filter computations for the blocks 5, 6, 7, and 8 are performed. For the coding tree block 1, the computation of blocks 4 and 7 are finished by now. The data for block 7 comes from buffer 1, and the data for blocks 5 and 6 comes from buffer 2. The data of block 8 comes from the currently active coding tree block 2.

Step 3. Block 10

For the computation of block 10 of the coding tree block 1, the register

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

array's working zone is located in the position r3 as shown in Fig. 6. At this position, it loads the data of block 9, 10, 11, and 12 into the register array's block A, B, C, and D respectively. Then the filter computations for the blocks 9, 10, 11, and 12 are performed. For the coding tree block 1, the computation of blocks 4, 7 and 10 are finished by now. The data for block 11 comes from buffer 1, and the data for block 9 and 10 comes from buffer 2. The data of block 12 comes from the currently active coding tree block 3.

Step 4. Block 13

For the computation of block 13 of the coding tree block 1, the register array's working zone is located in the position r4 as shown in Fig. 6. At this position, it loads the data of block 13, 14, 15, and 16 into the register array's block A, B, C, and D respectively. Then the filter computations for the blocks 13, 14, 15, and 16 are performed. For the coding tree block 1, all the computation of sub-block 4, 7, 10, and 13 are finished now. The data for block 15 comes from buffer 1, and the data for block 13 and 14 comes from buffer 2. The data of block 16 comes from the currently active coding tree block 3.

From this example, we can understand that the computation of one coding tree block (coding tree block 1 in this case) is completed gradually as the currently active coding tree block changes from 1 to 4, and the register array's working zone changes from r1 to r4 accordingly.

4 Performance analysis

The performance of the proposed architecture is analyzed as follows. For the computation of one 8×8 coding tree block in the register array, it takes 16 clock cycles. One frame of 4 K UHD can consist of 196,608 coding tree blocks of 8×8 in the worst case (including the Cb and Cr frames). There are supposed to be many coding tree blocks larger than 8×8 in general. However, we assume the worst case scenario for a robust design. For the frame rate of 30 fps, the number of the 8×8 coding tree blocks becomes 5,898,240. Since it takes 16 clock cycles to process one coding tree block, 94.4 million clock cycles are required to process the 4 K UHD at 30 fps. We need to run the architecture at 94.4 MHz to meet the performance. The size of the proposed register array is 3 K gates. The size of the hardwired logic including the filters is 43 K gates. The size of on-chip memories is 8 K gates. The total gate count of the proposed loop filter is 54 K gates. The operating clock frequency is 225 MHz in TSMC 65 nm technology. We can process the video data 2.38 times as large as that of 4 K UHD at 30 fps.

Even though the computation module meets the performance requirement, the I/O bandwidth must meet the performance requirement too. If the bus is 64-bit wide, it delivers the data of 1.8 GB/sec at 225 MHz. We mentioned that the I/O data bandwidth of the loop filter is more than

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013

750 MB/sec. The current coding tree block can be transferred from the prediction module by a separate dedicated channel without using the system bus. Then we can subtract 377.5 MB/sec of the coding tree block input from 750 MB/sec. The resulting I/O bandwidth required is 372.5 MB/sec. This is 18.6% of the total bus bandwidth. Depending on the bandwidth requirement of other modules of H.265 codec, or the multimedia SoC employing the H.265 codec, the bus size can be changed to meet the bandwidth requirement.

The performance of the proposed architecture can be easily scaled up by using more filters to compute in parallel. The connection between the registers needs to be modified for the shifting of multiple rows and columns. The degree of parallelism for filter computation can be increased up to 4 without increasing clock frequency. At 225 MHz, the maximum performance with the full parallelism leads to the performance to compute 8 K UHD at 60 fps.

5 Conclusion

In this paper, we presented the VLSI design of loop filter for H.265/HEVC. A novel parallel architecture employing register array was introduced. The register array consisted of 8×8 registers with two data flow directions. With the proposed register array and the on-chip SRAMs, we could compute the 4 K UHD at 30 fps in real-time. The parallel architecture was easy to scale up for higher performance to handle the video data up to 8 K UHD at 60 fps.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number 2012-0007468).

DOI: 10.1587/elex.10.20130161
Received March 05, 2013
Accepted March 18, 2013
Published April 10, 2013