doi:10.3772/j.issn.1006-6748.2022.04.007

A simplified hardware-friendly contour prediction algorithm in 3D-HEVC and parallelization design①

JIANG Lin(蒋 林)[**], DUAN Xueyao②[*], XIE Xiaoyan[***]
([*] College of Safety Science and Engineering, Xi'an University of Science and Technology, Xi'an 710054, P. R. China)
([**] Laboratory of Integrated Circuit Design, Xi'an University of Science and Technology, Xi'an 710054, P. R. China)
([***] School of Computer, Xi'an University of Posts and Telecommunications, Xi'an 710121, P. R. China)

Abstract

After the extension of depth modeling mode 4 (DMM-4) in 3D high efficiency video coding (3D-HEVC), the computational complexity increases sharply, which causes the real-time performance of video coding to be impacted. To reduce the computational complexity of DMM-4, a simplified hardware-friendly contour prediction algorithm is proposed in this paper. Based on the similarity between texture and depth map, the proposed algorithm directly codes depth blocks to calculate edge regions to reduce the number of reference blocks. Through the verification of the test sequence on HTM16.1, the proposed algorithm coding time is reduced by 9.42% compared with the original algorithm. To avoid the time consuming of serial coding on HTM, a parallelization design of the proposed algorithm based on reconfigurable array processor (DPR-CODEC) is proposed. The parallelization design reduces the storage access time, configuration time and saves the storage cost. Verified with the Xilinx Virtex 6 FPGA, experimental results show that parallelization design is capable of processing HD 1080p at a speed above 30 frames per second. Compared with the related work, the scheme reduces the LUTs by 42.3%, the REG by 85.5% and the hardware resources by 66.7%. The data loading speedup ratio of parallel scheme can reach 3.4539. On average, the different sized templates serial/parallel speedup ratio of encoding time can reach 2.446.

Key words: depth modeling mode 4 (DMM-4), contour prediction, 3D high efficiency video coding (3D-HEVC), parallelization, reconfigurable array processor

0 Introduction

The traditional high efficiency video coding (HEVC)[1] standard is ideal for exploring features of texture views[2], using the HEVC to encode depth map may cause significant distortion at the edges of the depth image[3]. 3D high efficiency video coding (3D-HEVC) uses multi-view video plus depth (MVD) format and the techniques such as depth image-based rendering (DIBR) to synthesize virtual views[4-5]. In order to improve the encoding quality and performance, 3D-HEVC introduced new tools——depth modeling modes (DMMs), DMM-1——DMM-4[5]. DMM-4 can better retain the edge information, which effectively solved the phenomenon of discontinuous object edge prediction, improving the quality of the composite video[6]. However, the computational workload of video coding increases dramatically, which makes the real-time processing by 3D-HEVC face great challenges. Therefore, it is urgent to propose an efficient solution to reduce computational complexity of contour prediction algorithm. Based on this, experts proposed some solutions for the DMM-4.

Ref.[7] introduced a fast decision of depth map coding based on decision tree. Each decision tree is used to decide when DMM evaluation can be avoided in depth map coding. Ref.[8] introduced a fast depth map internal prediction mode selection process based on machine learning and self-organizing map to skip unnecessary depth map internal prediction mode. According to the gradient information, Ref.[9] proposed a fast mode decision algorithm.

Due to the large number of reference pixels and complex calculation mode of DMM-4, the existing schemes cannot select or skip DMM-4 under certain condition. However, in practical application, DMM-1

① Supported by the National Natural Science Foundation of China (No. 61834005, 61772417, 61802304, 61602377, 61874087, 61634004) and the Shaanxi Province Key R&D Plan (No. 2020JM-525, 2021GY-029, 2021KW-16).
② To whom correspondence should be addressed. E-mail: duanxueyao@stu.xust.edu.cn.
Received on Aug. 24, 2021

mode cannot effectively deal with the coding at the boundary of a large number of objects and backgrounds. The hardware architecture has the characteristics of higher performance and lower power consumption, some experts have designed the hardware architecture for DMM-4 algorithm.

Ref. [10] presented a dedicated hardware architecture for the DMM-4 of 3D-HEVC emergent standard. This architecture encoded all available block sizes in parallel. But this architecture designed can't meet the new introduction for 3D-HEVC. Ref. [11] designed a real-time scalable hardware architecture, for which both DMM-1 and DMM-4 work. Still, this work is not fully compliant with the latest 3D-HEVC standard. Ref. [12] proposed a low-power and high-throughput architecture for the DMM-4. It achieved the higher processing rate in the comparative work, using fewer logical elements and registers. Ref. [13] proposed a hardware solution reducing the DMMs complexity. The solution contained five modules working in parallel. Ref. [14] extended the related research to build another architecture to encode bipartition modes. This hardware design included DMM-1 and DMM-4, and achieved different throughput according to application requirements. Ref. [15] proposed some simplifications that remove the less significant prediction modes and block sizes and dedicate hardware architecture for the 3D-HEVC depth map intra-prediction.

Real-time 3D video coding is a task with high computational cost and high processing speed, so it is necessary to optimize the contour prediction algorithm, and the efficient hardware architecture is also needed to meet the performance and resource requirements. The reason for the high computational complexity of DMM-4 is that texture map needs to be introduced as a reference when coding depth map, the amount of reference blocks increases significantly. And the serial calculation method of different sized templates causes time consumption. To solve these two problems, this paper mainly did the following work.

Through the analysis of the structural correlation between texture map and depth map at the contour between object and background, this paper proposes a simplified contour prediction algorithm. The proposed algorithm reduces the number of texture reference blocks, thus reducing the computational complexity of encoding. The DMM-4 algorithm employs the quad-tree coding structure that divides the depth maps into different sized coded tree units (CTUs), which is highly adapt to the multi-core parallel array structure of DPR-CODEC[16] designed by our project team. Therefore, this paper proposes a parallelization design of proposed algorithm based on DPR-CODEC to improve the encoding efficiency.

Experiments show that the proposed algorithm saved encoding time by 9.482%. Compared with the related work, this work reduced the hardware resources by 66.7%. The speedup ratio of parallel scheme of data loading can reach 3.4539, and the serial/parallel speedup ratio of encoding time can reach 2.446.

The rest of this paper is organized as follows. The simplified contour prediction algorithm and experimental results are introduced in Section 1. DPR-CODEC hardware structure and the parallelization design are explained in Section 2. The experiment of parallelization design is carried out in Section 3. Conclusion is given in Section 4.

1 Related work

1.1 DMM-4 algorithm in 3D-HEVC

In DMMs, the depth prediction block is divided into two regions. DMMs have wedgelet and contour partition methods, Fig. 1 shows two examples of wedgelet and contour partition. DMM-4 is the only mode in DMMs that uses contour partition. Contour partition uses two regions with arbitrary shapes to divide a PU into two regions, and there is no matching template for such edge partition method in DMM-1. Each region is represented by a different constant partition value (CPV).

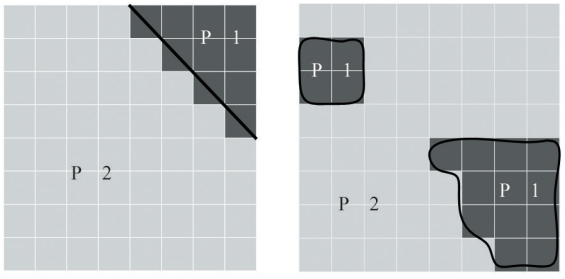


Fig. 1 Example of DMM-1 (a) and DMM-4 (b) partition

The main idea of DMM-4 contour prediction is to predict contour partition from texture reference block. In DMM-4, the texture map is used as a reference to encode the depth block. DMM-4 uses a threshold criterion in partition prediction. The threshold used in this example is the average value of texture blocks associated with coded depth mapping blocks, as shown in Fig. 2(a). The regions are divided as follows: the samples with texture blocks larger than the threshold are marked as Region 1, and the samples with texture blocks smaller than the threshold are marked as Region 0, as show in Fig. 2(b). Then, the average values of

Region 1 and Region 0 are used as the predicted values of each region, which are denoted as CPV 1 and CPV 0, as shown in Fig. 2(c). Finally, the original sample is subtracted from the predicted sample to produce residues, as shown in Fig. 2(d).

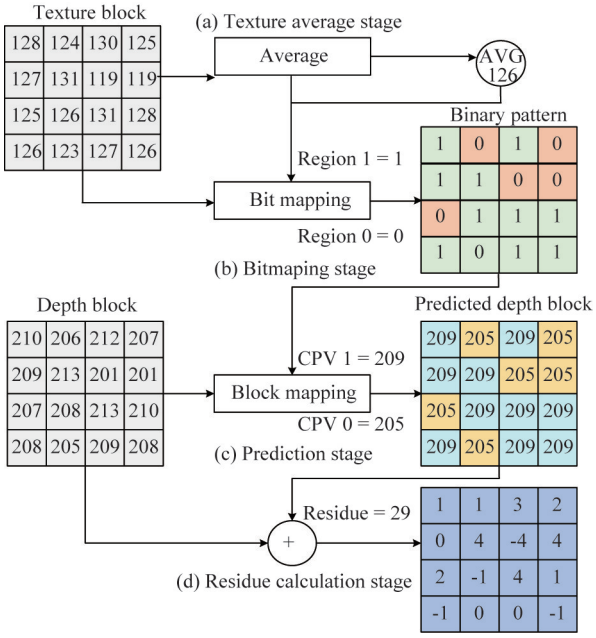


Fig. 2 DMM-4 encoding in HEVC

By testing the different test sequences on the 3D-HEVC standard test platform, the proportion of depth map encoding in the 3D-HEVC encoding process is shown in Fig. 3. In the entire encoding process, texture map encoding accounts for 52.5% of the entire encoding process, while depth map accounts for 47.50%. Depth mapping occupies nearly half of the encoding occupancy rate, and DMM-4 accounts for 3.34%. The introduction of DMM-4 brings an increase in the encoding time. When the DMM-4 algorithm encodes and predicts the current depth block, it not only encodes the current depth blocks, but also introduces the texture blocks in the same scene as references. The number of reference blocks doubles compared with other prediction modes. As a result, the real-time performance of video coding is damaged. However, due to

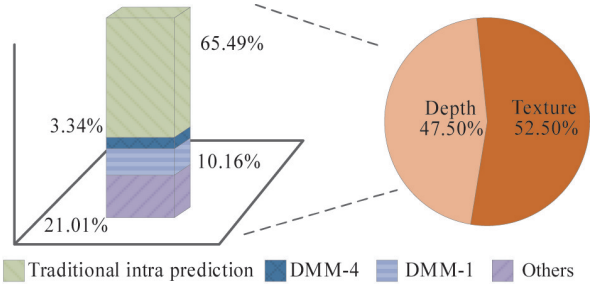


Fig. 3 Proportion of depth map encoding time

the ability to retain the encoding at the edge of the depth image, DMM-4 is critical and indispensable in 3D-HEVC. This paper analyzes the coding method of DMM-4 and the correlation between depth map and texture map, then proposes a simplified contour prediction algorithm.

1.2 A simplified contour prediction algorithm

Depth images and texture images express information from different angles. As shown in Fig. 4(a), texture images can clearly express objects and bring better visual effects. As shown in Fig. 4(b), the depth image has the characteristics of smooth interior of the object and sharp edge between the object and the background. However, in the same scene, texture map and depth map have similar structures. As shown in Fig. 4(c) and Fig. 4(d), depth map is not as sharp as texture map visually, but both can clearly express the boundary information between the object and the background, meeting the computational requirements of edge detection of the contour prediction algorithm. In addition, the traditional algorithm takes texture map as reference block, which not only increases the computational complexity, but also interferes with edge detection due to the excessive fine expression of texture image. As shown in Fig. 4(c), in the traditional DMM-4 algorithm, the bright spots of texture map are detected as the edge of the object for contour division, resulting in inaccurate contour prediction results. Therefore, using depth map as the reference block during the threshold calculation can meet the contour division requirements

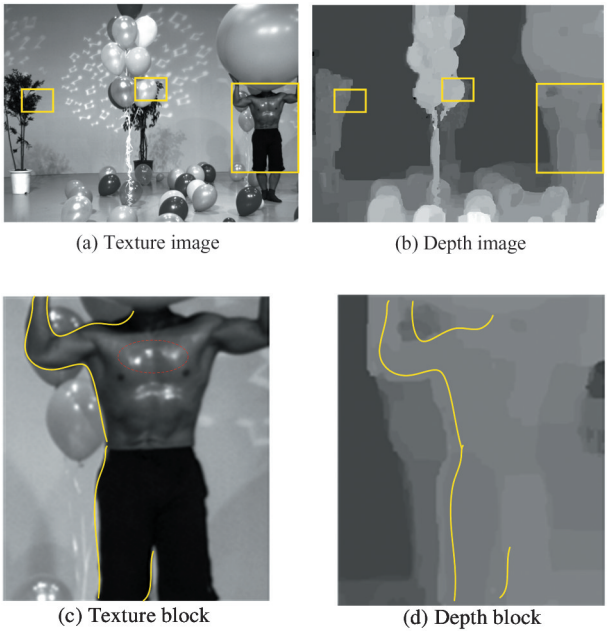


Fig. 4 Correlation between texture and depth image

of the algorithm. Based on above analysis, a simplified contour prediction algorithm is proposed in this paper.

During threshold calculation stage, the average value of the depth blocks instead of the texture reference blocks is directly calculated, as shown in Fig. 5(a). The average value is calculated as in Eq. (1).

$$average\_value = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{P}(i,j)}{N^2} \quad (1)$$

where $average\_value$ is the threshold of current depth block, N is the block size, $\boldsymbol{P}(i,j)$ is the pixel value, i and j represent the coordinates of the pixel in the discrete matrix, $0 < i < N+1$, $0 < j < N+1$. After that, the depth threshold is compared with the original depth pixels. If $\boldsymbol{P}(i,j) > average\_value$, the mapping value of point $\boldsymbol{P}(i,j) = 1$, otherwise it is 0, then the regions of the depth image are divided, as shown in Fig. 5(b).

Then calculate the predicted depth blocks. Calculating the averages of the Region 1 and Region 0 to obtain CPV 1, CPV 0, and map the values of CPV 1, CPV 0 to the corresponding regions to obtain the predicted depth blocks, as shown in Fig. 5(c).

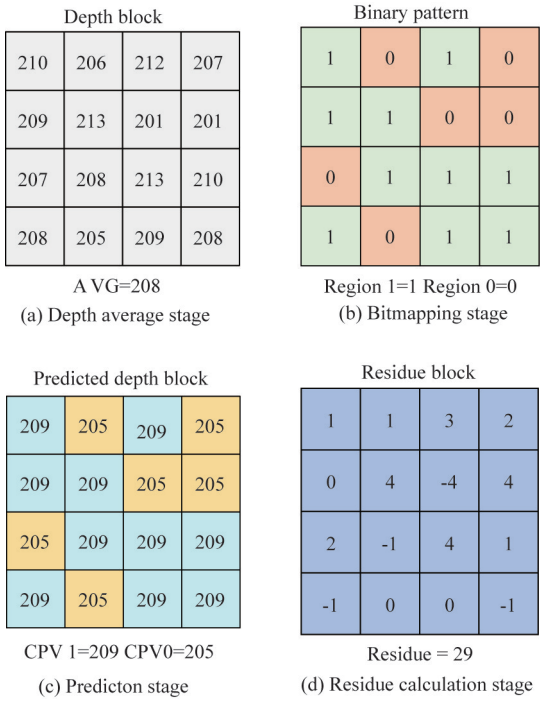


Fig. 5 Simplified contour prediction coding example

Last, perform residue calculation. The sum of absolute difference (SAD) is calculated between the original depth block and the predicted depth block, and the residue matrix is obtained, as shown in Fig. 5(d). The SAD matrix is summed to obtain the final residues. The $\boldsymbol{SAD}(i,j)$ value is calculated according to Eq. (2).

$$\boldsymbol{SAD}(i,j) = \sum_{i,j} |\boldsymbol{S}_A(i,j) - \boldsymbol{S}_B(i,j)| \quad (2)$$

where $\boldsymbol{S}_A(i,j)$ is the original pixel value of the depth map and $\boldsymbol{S}_B(i,j)$ is the predicted value of the depth map.

1.3 Effect of simplified contour prediction

This section introduces the coding efficiency and coding effect of the simplified contour prediction algorithm. In order to analyze the feasibility of the algorithm, timing tests and quality tests are simulated on the Matlab. Image compression and image reconstruction quality are measured by peak signal-to-noise ratio (PSNR). PSNR reflects the accuracy of the video coding algorithm, the higher the value, the higher the accuracy, and the smaller the resulting video distortion.

The simplified contour prediction algorithm focuses on shortening coding time by reducing the number of reference blocks. Based on the calculation, the reference blocks can be effectively reduced to 1/2 of the original algorithm, decreasing the number of reference blocks by 50%. To verify the effectiveness of the proposed algorithm, the five standard test sequences of HTM, Balloons, GT _ FLY, Newspaper, Poznan _ Hall2, Poznan _ Street, are tested and compared. The PSNR results of the simplified contour prediction algorithm compared with HTM16.1 are shown in Table 1. It is noted that the average loss of PSNR is 6.6471 dB. This figure is slightly higher. However, the simplified contour prediction algorithm proposed in this paper mainly solves the problem that the real-time performance of video coding is impaired due to the high computational complexity of DMM-4 algorithm. The simplified algorithm can significantly reduce the computational complexity of contour prediction algorithm. And the PSNR loss in GT _ Fly and Newspaper are reduced to 4.7304 dB and 5.1881 dB. These results show that the coding quality of the simplified contour algorithm is acceptable.

Table 1 PSNR of simplified contour prediction algorithm compared with HTM

Test sequences	HTM16.1 PSNR/dB	This paper PSNR/dB	△PSNR/dB
Balloons	42.1804	34.5173	-7.6631
GT _ FLY	52.8759	48.1455	-4.7304
Newspaper	43.0144	37.8263	-5.1881
Poznan _ Hall2	53.2603	43.7656	-9.4947
Poznan _ Street	50.2640	44.1049	-6.1591
Average	48.3190	41.6719	-6.6471

ΔT represents the depth map coding time variation of the proposed algorithm compared with the original algorithm, ΔT is defined as

$$\Delta T = \frac{T_{\text{original}} - T_{\text{proposed}}}{T_{\text{original}}} \times 100\% \quad (3)$$

where T_{original} is the DMM-4 coding time in HTM, and T_{proposed} denotes the depth coding time of algorithm proposed in this paper. To test the time saving of simplified contour prediction algorithm, five test sequences of HTM standard, Balloons, GT _ FLY, Newspaper, Poznan _ Hall2, Poznan _ Street, were tested and compared under common test condition (CTC). The encoding time of different sizes templates is compared with the original algorithm on HTM16.1, and results are shown in Table 2. Through the analysis of Table 2, it is obvious that the improvement reduced the depth map coding time by 9.482% on average, which proved the feasibility and effectiveness of the simplified contour prediction algorithm that the proposed in this paper.

Table 2 Depth coding time comparison under CTC

	Balloons	GT _ Fly	Newspaper	Poznan _ Hall2	Poznan _ Street	Average
4×4	7.55	7.69	7.81	6.45	6.78	7.256
8×8	8.93	10.91	8.33	9.38	11.86	9.882
16×16	10.88	12.07	11.86	9.43	12.31	11.31
ΔT/%	9.12	10.22	9.33	8.42	10.32	9.482

In 3D-HEVC, the contour prediction algorithm divides the depth map according to the different size templates in turn, the 8×8, 16×16 templates need to wait for the completion of the division and calculation of the 4×4 templates. From Table 2 it can be seen that in the same test sequence, ΔT increases with the increase of template size. Although the proposed algorithm reduces the reference blocks, it still inherits the serial coding method of traditional contour prediction. When the template size is small, the template will divide the encoded image into more layers of sub-blocks. While the sub-blocks of the same size template are divided and calculated, the calculation data of other different sub-blocks also need to wait for each other. This results in the longest time consumption of 4×4 template in contour prediction coding. To avoid time consumption in video coding, this paper proposes a parallel design for different sizes templates and sub-blocks of same size templates.

2 Parallel realization of simplified contour prediction

This section adopts the idea of data-level parallelism, and proposes a parallelization design hardware architecture for the simplified contour prediction algorithm based on the DPR-CODEC. The parallelization design uses different PEs to process different sized PUs concurrently to avoid data waiting.

2.1 Hardware architecture of DPR-CODEC

The DPR-CODEC supports H.264/AVC, H.265/HEVC and other video codec standards. The structure diagram of the reconfigurable array processor is shown in Fig. 6, which mainly includes a global controller, a two-dimensional adjacent interconnected reconfigurable array, data input memory (DIM), and data output memory (DOM). The hierarchical programming network is designed to load instructions through the H-tree network. Among them, as the core control unit of reconfigurable processing, the global controller is responsible for configuration information management, array state collection and data interaction. The reconfigurable array structure mainly completes the data calculation and processing work. 64 logical clusters processor element groups (PEGs) are

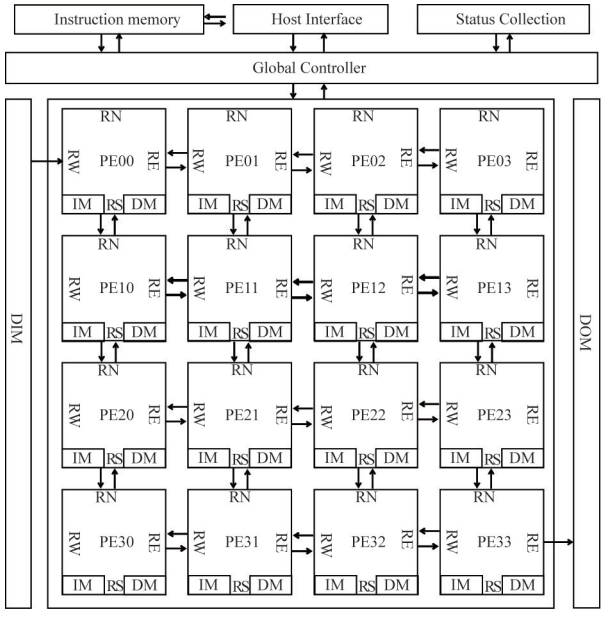


Fig. 6 Hardware architecture of DPR-CODEC

arranged into an 8 × 8 two-dimensional adjacent interconnected square array. Each PEG has the same internal structure. A PEG is mainly composed of 4 × 4 PEs, and uses adjacent interconnection for data interaction, including 2 kB instruction memory (IM) and 1 kB data memory (DM). In the structure, PEs of the same structure can complete multiple application functions according to different configuration information.

In the parallelization design hardware architecture proposed in this paper, the depth image is stored in an external memory. The PEs load the data from the external storage into the local data memory through DIM and feeds it back to the global controller. The global controller sends the corresponding instruction to the instruction memory of the PEs, and the PEs perform the next coding operation according to the instruction.

2.2 The design of parallelization

3D-HEVC adopted a flexible quad-tree coding structure and used serial encoding method. Although the child CU inherits the parent CU prediction mode to reduce the iterations in the HTM, the complex iterative calculations and serial encoding method consume most of the encoding time. DPR-CODEC is a multi-core dynamic reconfigurable video array processor with natural parallel structure. Since 3D-HEVC basically processes data in $N \times N$ rectangular blocks, DPR-CODEC can better meet the calculation requirements of the proposed algorithm. Due to the simplified design of proposed algorithm, now the contour prediction algorithm is more friendly to hardware realization. The simplified algorithm can reduce the visiting storage time and configuration time during the hardware implementation and reduce the consumption of hardware resources.

The PE array of parallelization design based on DPR-CODEC is shown in Fig. 7. The DIM is used for the data cache, such as the YUV video sequence data. PE00 is used to read the block data, template segmentation, PE30 is used to assign the data to specific function modules. PE01, PE02, PE31, PE32 are used to process the threshold calculation, bitmapping, CPV

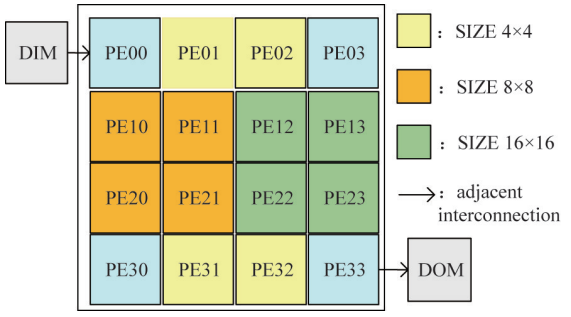


Fig. 7 Simplified contour prediction algorithm parallelization design array structure partition diagram

calculation and depth block prediction process of the 4 × 4 templates. PE10, PE11, PE20 and PE21 handle 8 × 8 templates, PE12, PE13, PE22 and PE23 handle 16 × 16 templates. PE03 is used for SAD calculation of each PU and the optimal template selection. The DOM is used to output the data. The specific parallelization design process is as follows.

Step 1 Data loading

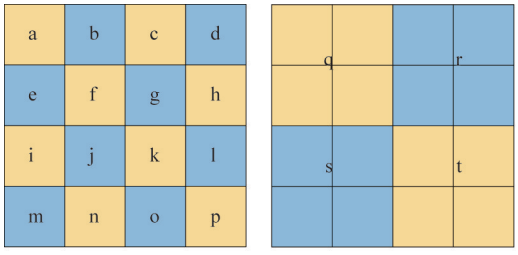
PE00 reads a 16 × 16 depth block from the DIM and stores the data in PE00 data memory address #0 – 255. After data is loaded, the depth information of the current block is transferred to PE30 in a shared storage manner.

Step 2 Data distribution and PU division

4 × 4 division: the data memory of PE01 receives depth data. The 256 depth pixels have been divided into 16 4 × 4 PUs, as shown in Fig. 8(a). PE01 loads blocks e, f, g, h to PE02, blocks i, j, k, l to PE31, and blocks m, n, o, p to PE32 in a way of shared storage manner. Blocks a, b, c, d remain in PE01.

8 × 8 division: data memory of PE10 receives depth data. The 256 depth pixels have been divided into 4 8 × 8 PUs, as shown in Fig. 8(b), PE10 loads block r to PE11, block s to PE12, and block t to PE21. Block q remains in PE10.

16 × 16 division: data memory of PE12 receives depth data.



(a) 4×4 block parallel scheme (b) 8×8 block parallel scheme

Fig. 8 PU division parallel scheme diagram

Step 3 Threshold calculation and bitmapping

4 × 4 coding: PE01, PE01, PE31, PE32 process the 4 × 4 coding blocks, the address #0 – 15 is the data address of the first block a. Click from top to bottom and from left to right. Similarly, #16 – 31, #32 – 63, #64 – 95 are the addresses of the last three blocks b, c, d. Calculate the average _ value of each block, where $N = 4$. After the threshold calculation is completed, then a loop is used to traverse the current depth block. If $\boldsymbol{P}(i, j) > average\_value$, $\boldsymbol{P}(i, j) = 1$, otherwise it is 0. It should be noted that each PE needs to perform bitmapping for four 4 × 4 blocks. The threshold calculation and bitmapping of 8 × 8 and 16 × 16 blocks are calculated in the same way.

Step 4 Depth block prediction

4 × 4 prediction: the same traversal method is used to perform a one-to-one correspondence between the bitmapping result in the previous step and the corresponding original depth block. The area corresponding to Region 1 is summed and averaged, the CPV 1 is the predicted value. Similarly, the area corresponding to Region 0 is summed and averaged to obtain CPV 0. The 8 × 8 and 16 × 16 blocks are predicted in the same way.

Step 5 Residue calculation

The optimal prediction block is judged according to the residues, and the smaller the residue shows that the division mode of the current block is the optimal one. The prediction value corresponding to this method is the optimal prediction value. According to the SAD values of the 4 × 4, 8 × 8, and 16 × 16 encoding blocks calculated in each PE, the optimal prediction mode is selected by computational comparison and the residue matrix is output through the DOM.

3 Experiment results and discussion

In order to test the performance of the proposed parallelization design of the simplified contour prediction algorithm, this section introduces the functional simulation platform and hardware implementation tools, also analyzes the comparison of the hardware implementation results with related work.

3.1 Hardware implementation

The implementation of a simplified hardware friendly contour prediction algorithm parallelization design based on DPR-CODEC design is as follows.

Firstly, the simplified contour prediction algorithm is programmed according to the special assembly instructions proposed by our project team. The special instruction translator is used to convert the assembly generation instructions into binary machine code and store them in the instruction memory of each PE of the DPR-CODEC. And the test data is preprocessed, and then put into the data memory of PE. Then, the function simulation of parallelization design is verified based on DPR-CODEC IDE. DPR-CODEC IDE is an integrated development environment based on DPR-CODEC hardware structure and video codec algorithm hardware simulation. DPR-CODEC IDE simulation platform is built by Modelsim system level modeling language SimC and HDL language simulation platform, and the numerical values and waveforms can be viewed in Modelsim for verification of functional simulation. After the function simulation is completed, the logic synthesis is carried out based on Xilinx ISE development tool, and the gate netlist file is generated. Finally, the corresponding bit file is generated by BEE4 platform, and the FPGA is configured to open the platform through BEE4 console to test.

3.2 Performance analysis and comparison

The parallelization design of the simplified contour prediction algorithm takes full advantages of the natural parallel structure characteristics and adjacent interconnection structure of DPR-CODEC for data loading and parallelizing the coding calculation of modules of different sizes. In order to verify the hardware performance of the parallelization design, this paper integrates the ISE 14.7 development environment of Xilinx and selects the BEE4 development board of BEEcube for FPGA verification. The experiment results are as follows.

This paper proposes to perform simultaneous coding for coding templets of different sizes to select the optimal templet. In order to facilitate the final optimal SAD, the PUs with sizes of 4 × 4 and 8 × 8 need to be synthesized into one 16 × 16 templet after the prediction of each PU is completed. Considering the overall reset characteristics of the array structure, the relevant data of all coding sizes load with the largest PU (16 × 16) is given. In order to effectively verify the accuracy of the parallel scheme (including data parallel scheme) designed in this paper, the serial data loading process of all coding units is integrated into the PE00 implementation to complete. Table 3 lists the 16 × 16 data loading time. It includes not only loading times for the 4 × 4, 8 × 8 and 16 × 16 templates, but also 159 592 clock cycles when PE00 reads data from DIM. According to the analysis, the data loading speedup ratio of parallel scheme can reach 3.4539.

Table 3 Serial/Parallel data loading time comparison (unit: clock cycle)

	Serial loading	Parallel loading	Speedup ratio
16 × 16	504 341	145 151	3.4539
DIM	159 592		-

In depth map encoding, our parallel design architecture adopts the idea of pattern parallelism to realize the parallel computation of 4 × 4, 8 × 8 and 16 × 16 templates. The encoding time of parallelization design on the DPR-CODEC of each size template is shown in Table 4. As can be seen from Table 4, the coding time of 4 × 4 template is the shortest, while that of 16 × 16 template is the longest. When the proposed algorithm is implemented on DPR-CODEC, every 4 PEs is used

to complete the coding calculation of 4 × 4, 8 × 8 and 16 × 16 template parallel processing. Since the 4 × 4 template can divide the depth block into more sub-blocks, the parallelism of data calculation can be higher in PEs, thus making the data call efficiency and coding efficiency higher. This correlation is the opposite of the serial encoding. The simplified contour prediction algorithm serial/parallel acceleration ratios of the 4 × 4, 8 × 8 and 16 × 16 code blocks are respectively 3.0148, 2.3753 and 1.9481. On average, the speedup ratio of encoding time can reach 2.446. According to the simulation results, the parallelization design based on the DPR-CODEC effectively improves the coding efficiency of the depth map contour prediction.

Table 4 Comparison of simplified contour prediction algorithm serial/parallel encoding time (unit: clock cycle)

Test sequences	Serial encoding	Parallel encoding	Speedup ratio
4 × 4		140 480	3.0148
8 × 8	423 525	178 301	2.3753
16 × 16		217 397	1.9481
Average	423 525	178 726	2.4460

Due to the reduction of reference blocks and the parallel scheme based on DPR-CODEC, the consumption of hardware resources is significantly reduced. The parallelization architecture designed in this paper required 31.7 K LUTs and 9.6 K REG, with maximum operating frequencies of 112.7 MHz.

To the best of our knowledge, only work[10-15] considers hardware solutions for DMM-4. In Table 5, the comparison of the synthesis results with Refs[11-16] is given. Ref. [15] and Ref. [12] synthesize for Nangate 45 nm, Ref. [14] synthesizes for ST 28 nm. The parallel architecture developed in this paper is synthesized for Xilinx Virtex 6 FPGA. Different synthesis processes are used in these three related work, which hampers the comparison with the synthesis results in this paper. Therefore, we can only evaluate the influence of the DMM-4 algorithm carried by hardware architecture on the encoding efficiency of intra-frame depth map prediction. Ref. [15] designed a hardware architecture for processing depth mapping and supports the complete 3D-HEVC intra-frame prediction model, but the architecture does not highlight the improvement on the intra-frame prediction algorithm itself. The hardware architecture that Ref. [15] designed can realize the encoding and decoding of DMM-1 and DMM-4, but this architecture only simplified the DMM-1 algorithm, did not improve the DMM-4 algorithm. In the above three works, the DMM-4 algorithm is not optimized in advance and implemented by serial in the hardware architecture. However, before designing the hardware architecture, this paper proposes a simplified hardware friendly contour prediction algorithm, which effectively reduces the data access time in the implementation of hardware structure. In addition, the parallel design scheme designed in this paper used 14 PEs parallel collaborative processing in DPR-CODEC, and adopts the method of data shared storage and parallel calculation of different templates, which improves the efficiency of contour prediction.

Ref. [13] and this paper are synthesized for Xilinx Virtex 6 FPGA, so it can be directly compared with this work. Compared with Ref. [13], the maximum frequency of the hardware architecture is slightly lower, but this work consumes less hardware resources. The LUTs in this paper is 42.3% less, the REG is 85.5% less, and the hardware resources are 66.7% less than the Ref. [13]. The architecture proposed by the Ref. [11] does not support 4 × 4 sized PU encoding. Compared with Ref. [11], LUTs and REG in this paper decrease by 61.9% and 90.3%, and the frequency of this work increase by 1.58 times compared with Ref. [11]. The Ref. [10] does not comply with the latest 3D-HEVC standard, although it adopts the parallel idea to propose a dedicated hardware architecture for DMM-4 algorithm. This architecture is not available for processing HD 1080p at a processing speed above 30 frames per second. Although less hardware resources are used in this Ref. [10], this work can

Table 5 Synthesis results comparison with the related work

Work	Technology	Area	REG	Maximum frequency/MHz	Throughput/fps
Ref. [15]	Nangate 45 nm	486.8 K gates	-	940	2160p@30
Ref. [14]	ST 28 nm	350.8 K gates	-	632.9	1080p@30
Ref. [13]	Xilinx FPGA Vrtex 6	55 K LUTs	67 K	275.0	2160p@30
Ref. [12]	Nangate 45 nm	139.84 K gates	-	750	2160p@30
Ref. [11]	Altera FPGA Startix V	83.3 K ALUT	100.2 K	71	1080p@30
Ref. [10]	Altera FPGA Startix V	5.6 K ALM	4.4 K	31.3	Not available
This work	Xilinx FPGA Virtex 6	31.73 K LUTs	9.66 K	112.72	1080p@30

achieve higher frequency. The maximum frequency of the hardware architecture proposed in this paper can reach 3.6 times of Ref. [10].

4 Conclusion

Aiming at the high computational complexity of DMM-4 algorithm and the time consuming problem caused by serial coding mode on HTM, this paper proposes a simplified contour prediction algorithm and a parallelization design of proposed algorithm based on DPR-CODEC. The proposed algorithm eliminates the dependency on texture reference blocks in threshold calculation of DMM-4. The reference blocks reduced by 50%. The coding time reduced by 9.482% compared with the original DMM-4 algorithm on HTM16.1.

In order to meet the real-time demands of video coding applications, this paper proposes a parallelization design based on DPR-CODEC to solve the time consuming problem of data waiting. The function simulation mapping is implemented on DPR-CODEC IDE, and the hardware performance is verified by BEEcube BEE4 FPGA hardware development platform. The parallelization design of the proposed algorithm benefits from the adjacent interconnect structure of DPR-CODEC, and the video data can be loaded between PEs in the way of shared storage, which saves the data access time and storage cost in the hardware structure. The serial/parallel acceleration ratio of data loading can reach 3.4539. Based on the natural parallel structure of DPR-CODEC, templates of different sizes are used for parallel computation. The waiting time of templates without data correlation is reduced and the computational efficiency is improved. The average serial/parallel acceleration ratio of 4×4, 8×8 and 16×16 templates can reach 2.446. Compared with other related work, our scheme reduced LUTs by 42.3%, REG by 85.5%, and hardware resource consumption by 66.7%.

References

[1] SULLIVAN G J, OHM J R, HAN W J, et al. Overview of the high efficiency video coding (HEVC) standard [J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2012, 22(12): 1649-1668

[2] FU C H, CHAN Y L, ZHANG H B, et al. Efficient depth intra frame coding in 3D-HEVC by corner points [J]. *IEEE Transactions on Image Processing*, 2020, 30: 1608-1622

[3] HAMOUT H, ELYOUSFI A. An efficient edge detection algorithm for fast intra-coding for 3D video extension of HEVC [J]. *Journal of Real-Time Image Processing*, 2019, 16(6): 2093-2105

[4] HAMOUT H, ELYOUSFI A. Fast depth map intra coding for 3D video compression-based tensor feature extraction and data analysis [J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019, 30(7): 1933-1945

[5] TECH G, CHEN Y, MÜLLER K, et al. Overview 1622 of the multiview and 3D extensions of high efficiency video coding [J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2015, 26(1): 35-49

[6] MERKLE P, MÜLLER K, MARPE D, et al. Depth intra coding for 3D video based on geometric primitives [J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2015, 26(3): 570-582

[7] MOURA C, SALDANHA M, SANCHEZ G, et al. Fast intra mode decision for 3D-HEVC depth map coding using decision trees [C]//2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, UK, 2020: 1-4

[8] HAMMANI A, HAMOUT H, ELYOUSFI A. Fast depth map intra mode prediction based on self-organizing map [C]//2020 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 2020: 1-5

[9] ZHANG Q, JING R, WANG B, et al. Fast mode decision based on gradient information in 3D-HEVC [J]. *IEEE Access*, 2019 (7): 135448-135456

[10] SANCHEZ G, ZATT B, PORTO M, et al. A real-time 5-views HD 1080p architecture for 3D-HEVC depth modeling mode 4 [C]// Proceedings of the 27th Symposium on Integrated Circuits and Systems Design, Aracaju, Brazil, 2014: 1-6

[11] SANCHEZ G, MARCON C, AGOSTINI L. Real-time scalable hardware architecture for 3D-HEVC bipartition modes [J]. *Journal of Real-Time Image Processing*, 2017, 13(1): 71-83

[12] ÜCKER M, AFONSO V, AUDIBERT L, et al. Low-power and high-throughput architecture for 3D-HEVC depth modeling mode 4 [C]//2018 31st Symposium on Integrated Circuits and Systems Design, Bento Gonçalves, Brazil, 2018: 1-6

[13] AMISH F, BOURENNANE E B. An efficient hardware solution for 3D-HEVC intra-prediction [J]. *Journal of Real-Time Image Processing*, 2019, 16(5): 1559-1571

[14] SANCHEZ G, SALDANHA M, FERNANDES R, et al. 3D-HEVC bipartition modes encoder and decoder design targeting high-resolution videos [J]. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2019, 67 (2): 415-427

[15] UCKER M, AFONSO V, SALDANHA M, et al. High-throughput hardware for 3D-HEVC depth-map intra prediction [J]. *IEEE Design and Test*, 2019, 37(3): 7-14

[16] SHAN R, JIANG L, WU H, et al. Dynamical self-reconfigurable mechanism for data-driven cell array [J]. *Journal of Shanghai Jiaotong University (Science)*, 2021, 26 (4): 511-521

JIANG Lin, born in 1970. He received his Ph.D degree from Xi'an Jiaotong University in 1996. He received his M.S. degree in Mechanical Engineering Department of Xi'an Jiaotong University in 1992. From 1998 to 2000, he engaged in postdoctoral research in the Aviation Microelectronics Center of Northwest University of Technology. He focuses on integrated circuit design, computer architecture, computer graphics and image processing technology. He has presided 2 National '863' Program Projects, 1 Key Project of National Natural Science Foundation of China, and more than 10 provincial and ministerial projects.