

# FPGA Architecture for Real-Time Ultra-High Definition Glasses-Free 3D System

Ran Liu<sup>1,2</sup>(✉), Mingming Liu<sup>1</sup>, Dehao Li<sup>1</sup>, Yanzhen Zhang<sup>2</sup>,  
and Yangting Zheng<sup>1</sup>

<sup>1</sup> College of Computer Science, Chongqing University, Chongqing 400044, China  
Ran.liu\_cqu@qq.com

<sup>2</sup> College of Communication Engineering, Chongqing University,  
Chongqing 400044, China

**Abstract.** This paper presents an FPGA architecture for real-time ultra-high definition (UHD) glasses-free 3D system by solving high bandwidth requirement and high complexity of the system problems. Video + Depth (V + D) video format and the depth-image-based rendering (DIBR) technique are supported by the system to reduce the requirement of bandwidth. In addition, an asymmetric shift-sensor camera setup is introduced to reduce the hardware cost as well as the complexity of the system. We also simplify the microlens array weight equations so as to reduce the complexity of subpixel rearrangement coefficients calculation for glasses-free 3D image creation. Experiments results show that the proposed architecture can support the resolution of 4K for real-time UHD glasses-free 3D display.

**Keywords:** Ultra high definition · Glasses-free 3D display  
Depth-image-based rendering · Multiview rendering · FPGA

## 1 Introduction

In recent years, Glasses-free 3D technology has gradually entered the public view. However, the lack of glasses-free 3D content and the shortage of transmission bandwidth for this content hinder the popularity of the glasses-free 3D system. To overcome these obstacles, the V + D video format begins to be applied to the system. Each frame in this video format consists of a 2D color image with a depth map, where the depth map is a gray image with the value varying from 0 to 255, which is usually obtained by depth estimation algorithm or depth camera. In addition, only one channel of video and its depth map are needed to transmit for this format, hence it requires lower bandwidth in comparison with the traditional 3D video. Once each frame has been received and decoded, the color and depth image are recovered and rescaled, and the virtual multiviews required for glasses-free 3D display can be generated from color and depth images by depth-image-based rendering (DIBR) technique [1, 2]. Finally, each display view is synthesized by image fusion method, and then the synthesized view is

displayed in a set of subpixels, each of which is viewable from a specific angle through a fine lenticular overlay placed over the LCD display [3].

However, this video format has some defects when it is used in a glasses-free 3D system. First, big holes may appear in the generated virtual view (destination image) after DIBR; hence high-performance hole-filling algorithm should be applied to fix these holes. Second, implementation of DIBR and hole-filling algorithm by software is time-consuming and cannot meet the requirement of real-time processing of video stream. In virtue of the power of field programmable gate array (FPGA) in real-time video processing [4], using FPGA to implement real-time glasses-free 3D system has become the consensus of current research. Until now, only a few studies focus on FPGA implementation of the real-time glasses-free 3D system. Most of them only discussed the traditional 3D video format, and the resolution of the 3D system is lower than ultra-high definition resolution, which cannot meet the requirement of the newest ultra-high definition display.

In this paper, we propose an FPGA architecture for real-time ultra-high definition glasses-free 3D system. We present the asymmetric shift-sensor camera setup to simplify the hardware implementation of multiview rendering module of the system. In addition, as this camera setup is applied, no DDR is needed for our FPGA architecture, hence the traditional scheme of using DDR for pipelining is avoided [5]. We also proposed a view fusion method which can rearrange the subpixels from the generated views to form a single glasses-free 3D image. The proposed architecture is implemented by Verilog HDL on the Xilinx Virtex-7 platform and simulated with ModelSim 10.2.

## 2 Asymmetric Shift-Sensor Camera Setup

As mentioned above, a glasses-free 3D system needs multiple views of the same scene to form a 3D image. In our system, these views are generated by DIBR technique using specified parameters in multiview rendering module. Note that the V + D video used in this paper has no camera calibration parameters, so it is necessary to modify the 3D image warping equations to adapt to the case without camera calibration parameters [6]. In this paper, a scheme of “fusing” 8 views into a glasses-free 3D image is adopted. Positions of these 8 viewpoints/virtual cameras determine the effect of glasses-free 3D display. As shown in Fig. 1,  $I_r$  is a reference image,  $U_r$  is a point on the reference image,  $U_r = [u_r, v_r, 1]^T \in I_r$ . The view of the same scene captured at different viewpoints is labeled as  $I_n (1 \leq n \leq 7)$ ; point  $U_i = [u_i, v_i, 1]^T \in I_n$ . Point  $U = [X_w, Y_w, Z_w, 1]^T$  (expressed in normalized homogeneous coordinates in the world coordinate system) is the intersection of  $u_r$  and  $u_i$  when they are mapped to the scene using the prejection matrices of  $I_r$  and  $I_n$ .  $B$  represents the length of the baseline between two adjacent virtual cameras (units: millimeter), Plane  $z_c$  ( $z_c > 0$ ) is the zero-parallax setting (*ZPS*) plane. By setting the plane  $z_c$ , we can generate a virtual with positive or negative parallax. Note that  $I_r$  is taken as one of the 8 views and placed on the rightmost. We call this camera setup asymmetric shift-sensor camera setup. Plane  $z_c$  is determined by (1).

$$z_c = \frac{f \cdot s_x \cdot B_n}{2h_n} \quad 1 \leq n \leq 7 \tag{1}$$

where  $S_x$  is the number of pixels per unit length in the  $x$  axis,  $f$  is the focal length of the camera,  $B_n$  represents the length between the camera corresponding to  $I_r$  and  $I_n$ ,  $B_n = n \times B$ .  $h_n$  denotes the horizontal sensor shift (measured in pixels) of  $I_n$  when setting the  $ZPS$  plane.

Note that the seven virtual views correspond to the same  $ZPS$  plane (plane  $z_c$ ), and their 3D image warping equations can be written as follows [2].

$$\begin{cases} u_i - u_r = (-1)^\alpha \left[ 2h_n - \frac{f \cdot s_x \cdot B_n}{z_w} \right] \\ v_i = v_r \end{cases} \quad 1 \leq n \leq 7. \tag{2}$$

where is defined as follows: if  $I_n$  is located on the right side of  $I_r$ ,  $\alpha = 0$ ; otherwise,  $\alpha = 1$ . As for our camera setup, all virtual views are located on the left side of  $I_r$ , hence  $\alpha = 1$ . Parameter  $z_w$  is the depth value of point  $u_r$ ; it is determined by:

$$z_w = \frac{1}{\frac{D(u_r, v_r)}{255} \times \left( \frac{1}{z_{min}} - \frac{1}{z_{max}} \right) + \frac{1}{z_{max}}} \tag{3}$$

where  $D(u_r, v_r)$  is the depth value of the point  $(u_r, v_r)$  in depth map,  $D(ur, vr) \in [0, 225]$ .  $z_{min}$  is the near clipping plane,  $z_{max}$  is the far clipping plane.

In the case of no camera calibration parameters, in order to simplify the calculation, the near shear plane  $z_{min}$  is usually set to 1, and the far shear plane  $z_{max}$  is set to infinity [6]. Therefore, Eq. (3) can be simplified as follows.

$$z_w = \frac{255}{D(u_r, v_r)} \tag{4}$$

Putting Eq. (1) into Eq. (2) yields the following equation:

$$\begin{cases} u_i - u_r = -n \cdot f \cdot s_x \cdot B \left( \frac{1}{z_c} - \frac{1}{z_w} \right) \\ v_i = v_r \end{cases} \quad 1 \leq n \leq 7. \tag{5}$$

Usually, the depth map stores depth information as 8-bit gray values with the gray level 0 specifying  $z_{max}$  and the gray level 255 defining  $z_{min}$  [7]. According to (4), (5) can be simplified to the following (6).

$$\begin{cases} u_i - u_r = -n \cdot p (D_{zps} - D(u_r, v_r)) \\ v_i = v_r \end{cases} \quad 1 \leq n \leq 7. \tag{6}$$

where  $p = f \cdot s_x \cdot B / 255$  is the scaling factor,  $D_{zps} = 255 / z_c$ ,  $D(u_r, v_r) = 255 / z_w$ .  $D_{zps}$  are depth value of  $ZPS$  plane,  $D_{zps} \in [0, 255]$ . According to [7, 8], the maximum horizontal sensor parallax  $d(u_i - u_r)$  which is comfortable for viewing is approximately 5% of the width of an image. Hence we have:

$$|d| \leq W \times 5\% \tag{7}$$

where  $W$  is the width of  $I_r$ , and both the units of  $d$  and  $W$  are pixels. Eq. (7) is applied to (6) to get (8).

$$0 \leq p \cdot n \cdot |D_{zps} - D(u_r, v_r)| \leq W/20 \quad 1 \leq n \leq 7. \quad (8)$$

$$0 \leq |D_{zps} - D(u_r, v_r)| \leq 255 \quad (9)$$

Therefore, the range of  $p$  value can be deduced from (8) and (9):  $0 \leq p \leq W/35700$ .

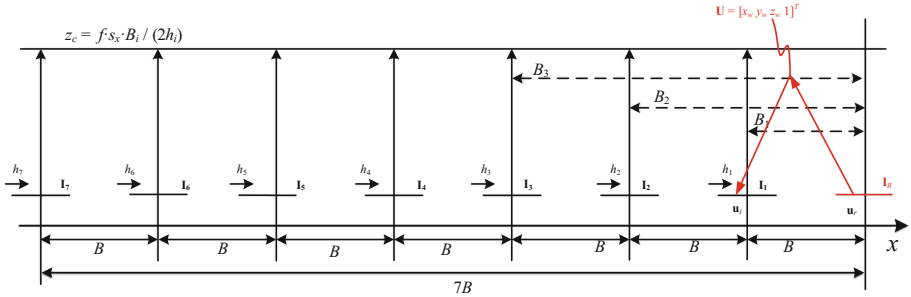


Fig. 1. Asymmetric shift-sensor camera setup

To facilitate hardware implementation, we introduce the integer arguments  $p'$ ,  $p' = p \times 2^{15}$ . Thus, Eq. (6) becomes as follows.

$$\begin{cases} u_i - u_r \approx -n \cdot \frac{p'}{2^{15}} (D_{zps} - D(u_r, v_r)) \\ v_i = v_r \end{cases} \quad 1 \leq n \leq 7. \quad (10)$$

Note the resolution of the ultra-high definition image is  $3840 \times 2160$ , hence  $W = 3840$ , the maximum possible value of  $p$  is  $\lceil 3840 \times 32768 / 35700 \rceil = 3524$ . When  $p$  is the maximum, the maximum absolute value of parallax  $d$  is  $\lceil 7 \times 3524 \times 255 / 32768 \rceil = 192$ , hence  $d \in [-192, 192]$ , which satisfies the requirement of Eq. (7).

### 3 Hardware Architecture

Our hardware architecture is based on Xilinx Virtex-7 FPGA development platform (XC7V585T). The overall system architecture is illustrated in Fig. 2. The system consists of 4 modules: video input, multiview rendering, view fusion and video output. Each module can be divided into several submodules. In multiview rendering module, the current row should be initialized before their parallax values are calculated. So the clock frequency of the system is chosen to reach more than twice the pixel clock frequency of 4K video. In order to reduce the

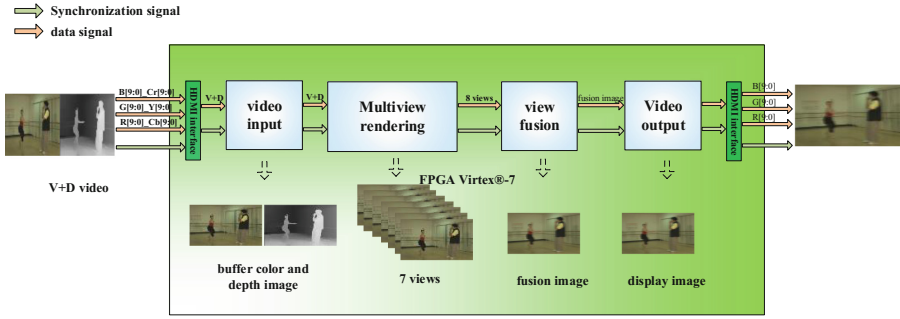


Fig. 2. Hardware architecture of the system

transmission bandwidth of the video,  $V + D$  video format begins is applied to the system. Each frame in  $V + D$  video is split, stretched and buffered in video input module. Video input module output a color image and its corresponding depth map whose resolutions are both  $3840 \times 2160$ . Multiview rendering module is composed of seven DIBR submodules that are responsible for generating seven different virtual views. View fusion module receives the reference image and the generated virtual views and synthesizes a glasses-free 3D image [9]. Video output module buffers the glasses-free 3D images and output them to the HDMI interface for display.

### 3.1 Video Input Module

The horizontal resolution of each frame in  $V + D$  format video is converted from 3840 to 7680 by video input module using column copy mode. Then, the reference image of each frame and its corresponding depth map is stored into different Block RAMs. In this way, the resolution of the reference image and depth map is expanded to  $3840 \times 2160$ . The architecture of video input module is shown in Fig. 3, which can be divided into system boot, stretching, splitting and buffer module.

Note that stretching module extends the input image data to twice the width of the original image using column replication, and then stores the data correctly into buff\_ram. Splitting module contains three Block RAMs: buff\_ram, col\_ram and dep\_ram, where buff\_ram's data input port and data output port bit width is 60 bits and 480 bits respectively. The width of data selected here should ensure that the output of data is expected to be completed when the data valid signal is negative. In addition, buff\_ram's input clock is the pixel clock, while its output clock is internal working clock of the system, so that the switch of clock domain is realized. Col\_ram and dep\_ram's input data bit width is 480 bits and its output data bit width is 30 bits and 8 bits respectively, which main function is to achieve the splitting of reference image and depth map. The data buffer module consists of four Block RAMs. By using three Block RAMs, the color data are delayed two lines by ping-pong operation and then entered into the subsequent DIBR module.

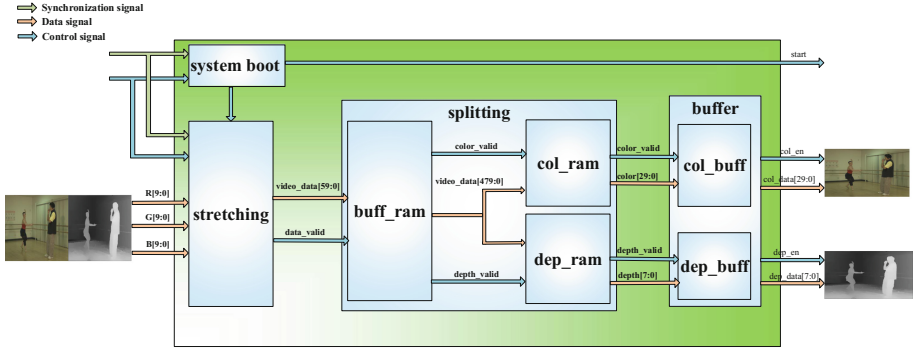


Fig. 3. Hardware architecture of video input module

### 3.2 Multiview Rendering Module

The multiview rendering module consists of 7 DIBR submodules, which are in charge of generating seven virtual views with different viewing angles. Therefore, this section will focus on DIBR working principle and its mainly design, as shown in Fig. 4. The DIBR module implements two algorithms: 3D image warping and hole filling algorithm mentioned in [10]. An asymmetric shift-sensor camera setup is achieved in 3D image warping. The reason for data valid signal extension is to complete the output of the last two rows of the disparity map after buffering. Small holes of virtual depth map are filtered by median filter in filter module. After hole filling, depth map without hole is used to generate novel view in pixel copy module. 3D image warping and hole-filling submodules are described in detail later.

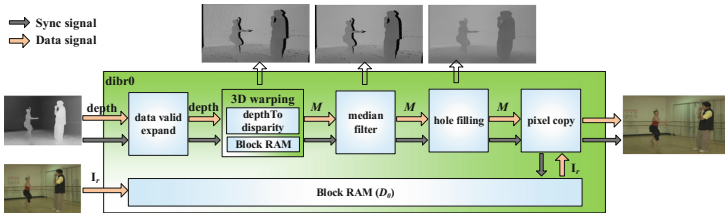
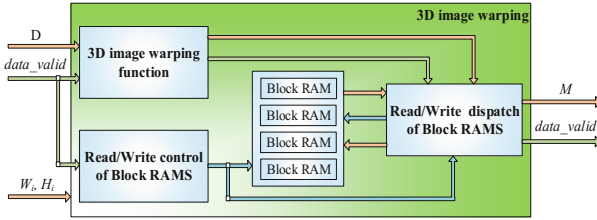


Fig. 4. Hardware architecture of DIBR

**3D Image Warping Submodule.** As shown in Fig. 5, the 3D image warping submodule receives depth value  $D$  outputted by data valid signal extension submodule, calculates the parallax value  $M$  of each pixel in the reference image according to  $D$ , and outputs  $M$  to disparity map filtering submodule. As three rows of parallax value are needed for median filtering, the 3D image warping submodule will buffer these data in the Block RAM (see Fig. 5), and output three parallax values in each clock to the median filter mask in disparity map filtering submodule. Three parallax values are read from the three rows respectively.



**Fig. 5.** Hardware architecture of 3D image warping submodule

Note that the destination image generated by 3D image warping may contain holes. In this paper, the parallax of a hole pixel is set to  $-256$  (the parallax cannot achieve at  $-256$  in order to avoid visual fatigue). In 3D image warping submodule, Block RAM is used to store parallax values which is initialized to  $-256$  before processing a depth map. Hence, the subsequent submodules will be able to distinguish which pixels are holes.

In 3D image warping submodule, the Look-Up-Table (LUT) *depth2disparity* is used to realize the depth-to-disparity conversion and it is implemented by ROM. The size of the LUT is  $256 \times 9$  bits, and the values stored in it are calculated in advance outside the system. These values are loaded by parameter setting submodule during the Vertical Blanking Interval (VBI). By this way, the function of (10) can be realized by the LUT *depth2disparity*. In addition, the depth adjustment can be achieved by changing the values in the *depth2disparity*. Another important aspect of the submodule is that the input sequence of the depth map is from right to left, top to bottom. In this way, occlusion-compatible results can be obtained with this sequence, therefore the folds can be eliminated without any extra hardware resources [11].

**Hole-Filling Submodule.** The hole-filling submodule receives the parallax value  $RX_M$  from disparity map filtering submodule, fills the holes in the disparity map, and then outputs it to the pixel-copy submodule. The hole-filling is implemented by modifying the parallax value of hole point (i.e.  $-256$ ) in the disparity map. According to the algorithm mentioned in [10], the hardware architecture of hole-filling submodule is implemented in Fig. 6. It is shown that hole-filling submodule consists of four function modules: detect\_hop, mark\_holes, fill\_holes and dilate\_big\_holes module.

The responsibility of detect\_hop module is distinguished from foreground and background. Then, the background parallax value of mutation is output. The mark\_holes module marks the different categories of holes in the data stream for the subsequent module to fill holes. At last, different methods are used to fill small and big holes in the fill\_holes and dilate\_big\_holes modules, respectively. The small holes are filled with adjacent pixels. After dilation, the big holes are filled with the background parallax value which comes from mutation of detect\_hop outputting.

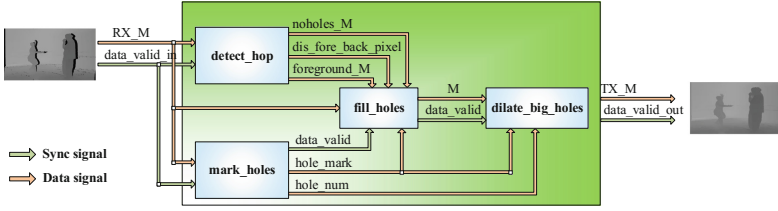


Fig. 6. Hardware architecture of hole-filling submodule

### 3.3 View Fusion Module

The view fusion module is responsible for subpixels rearrangement of the eight views. Then, subpixels in accordance with arrangement rules of the cylindrical lens glasses-free 3D displays are arranged to generate a stereoscopic image and sent to the HDMI interface [12]. In different glasses-free 3D displays, the subpixel arrangement requirements are different, even if the size of the screen is same. The subpixel arrangement of grating can be divided into integer arrangement and floating-point arrangement. In particular, the integer arrangement can be seen as a special case of floating-point arrangement. The model used in this paper support 4K floating-point cylindrical lens gratings. In addition, lenticular screens are typically slanted at a small angle to reduce a vertical banding effect known as the “picket fence” (an artifact of the microlenses magnifying the underlying LCD mask). This tilt angle is usually divided into two cases, counterclockwise and clockwise. This paper uses an anticlockwise tilt angle. Here we use the floating-point arrangement and counterclockwise tilt angle as an example to introduce the realization of view fusion module. Figure 7 illustrates a slanted lenticular layout with RGB stripe subpixels. According to the physical characteristics of 4K glasses-free displays, the RGB pixel values of the composite image are calculated using the formula (11), and the microlens array weights are calculated using (12) and (13). Note that (13) is further optimized on the basis of previous method in [9].

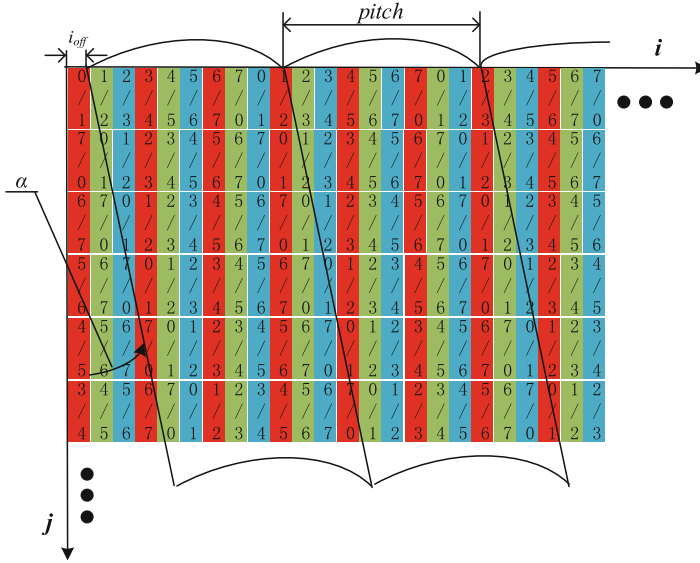
$$R_{i,j} = \sum_{n=1}^N F_{i,j}^n \times r_{i,j}^n \quad G_{i,j} = \sum_{n=1}^N F_{i,j}^n \times g_{i,j}^n \quad B_{i,j} = \sum_{n=1}^N F_{i,j}^n \times b_{i,j}^n \quad (11)$$

$$F_{i,j}^n = \begin{cases} 1 - (k_{i,j} - \lfloor k_{i,j} \rfloor) & \text{if } n = \lfloor k_{i,j} \rfloor \\ k_{i,j} - \lfloor k_{i,j} \rfloor & \text{if } n = \lfloor k_{i,j} \rfloor + 1 \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

$$K_{i,j} = \frac{(i - i_{off} - 3j \times \tan\alpha) \% pitch}{pitch} \times N \quad (13)$$

where  $i$  and  $j$  denote the horizontal and vertical coordinates,  $i_{off}$  is the horizontal offset,  $i_{off} = 0$ ,  $i \in [0, 3840 \times 3)$  and  $j \in [0, 2160)$ , because of RGB three channel.  $N$  denote the number of views ( $N = 8$ ),  $pitch$  denote the ratio of pixels per inch and per inch, which is a constant ( $pitch = 8.01$ ). Each pixel coefficient





**Fig. 7.** Slanted lenticular layout with RGB stripe subpixels: two different numbers in one rectangle denote the two participating subpixels from which viewpoint

of the composite image is calculated directly by (13), which greatly consumes hardware resources and increases the complexity of the hardware implementation. Therefore, we make an equivalent transformation to get (14), which has been verified by computer.

$$K_{i,j} = \frac{(i - i_{off} - 3j \times \tan\alpha) \times N}{pitch} \% N \tag{14}$$

By observing (14), it can be seen that once the RGB subpixel coefficients of the first pixel are determined, the subsequent subpixel coefficients can be obtained by adding or subtracting the constant from the previous subpixel coefficients. The subpixel coefficients calculation equation in the horizontal and vertical direction can be obtained respectively by (14).

$$K_{i+3,j} = (K_{i,j} + 3\rho) \% N \quad \rho = N/pitch \tag{15}$$

$$K_{i,j+1} = (K_{i,j} - 3\rho \tan\alpha) \% N \quad \rho = N/pitch \tag{16}$$

According to the above deduction, the coefficient  $K$  can be obtained by simple addition or subtraction and one modular operation, which greatly saves the resource consumption and reduces the technical difficulty of the implementation. In the design of this module, the first subpixel coefficients of each image are fixed constants, which can be stored in registers, such as  $K_{0,0} = 0$ ,  $K_{1,0} = 0.9988$ ,  $K_{2,0} = 1.9975$ ,  $\tan\alpha = 0.2344$ ,  $\rho = 0.9988$ . In addition, we adopted IEEE 754 to governs binary floating-point arithmetic, which can greatly guarantee the

accuracy of floating-point calculation. The view fusion module consist of three submodules: buffer module, coefficients calculate module and subpixel rearrangement module, and among which the most difficult is coefficients calculate module. Figure 8 demonstrates the coefficients calculation module at the circuit-level.

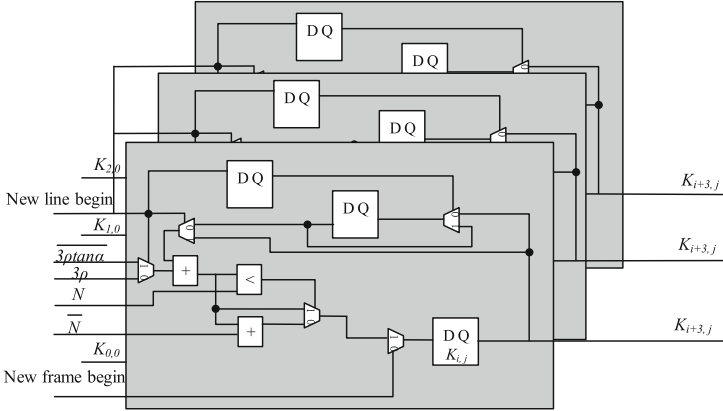


Fig. 8. Subpixel coefficients calculation circuit

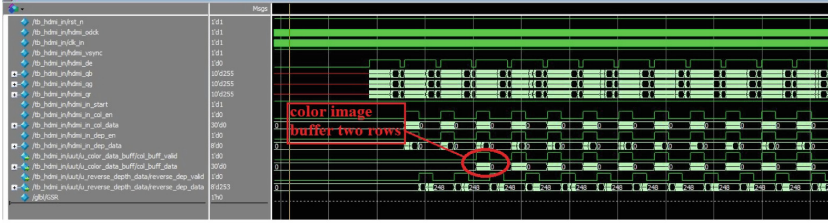
### 4 System Implementation

We used Verilog HDL to implement the design module of system. A proof-of-concept implementation was synthesized and simulated for a Xilinx XC7VX330T FPGA using Xilinx Vivado 2014 toolset and ModelSim 10.2. According to the HDMI 1.4, the 4K ultra-high definition video has a transmission rate of 297 MHz on the HDMI interface. Therefore, in view of the performance of the hardware platform and the processing efficiency of whole system, 666 MHz clock frequency was adopted in the system.

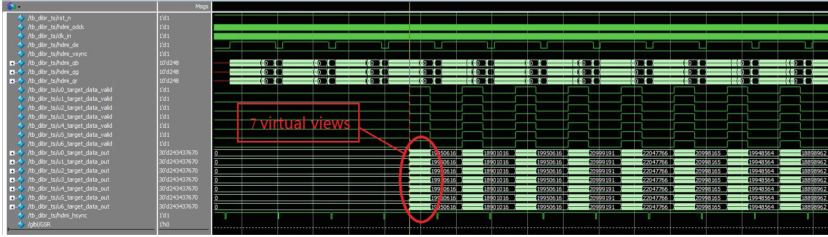
The timing sequence of video input module is shown in Fig. 9(a). We can see that the module implemented the splitting, stretching, and buffering of input image correctly. The reference image and depth map are output by 30 bits and 8 bits respectively.

Timing simulation of multiview rendering module is shown in Fig. 9(b). From this figure, we can see that the module has instantiated 7 DIBR submodules and produced expected timing.

Until now, few studies focus on 8 viewpoints real-time UHD glasses-free 3D system. Hence it is difficult for us to find an appropriate design for comparison. In this paper, we compared the symmetrical shift-sensor camera setup, which was proposed in our previous study [13], with asymmetric shift-sensor camera setup for multiview rendering module. Table 1 summarizes the detailed device utilization results. Compared with a previous design, our innovative architecture saves 1799 slice registers and 1526 slice LUTs for 4K@30 Hz video.



(a) Video input simulation



(b) Multiview rendering simulation

**Fig. 9.** Timing simulation of main submodules;

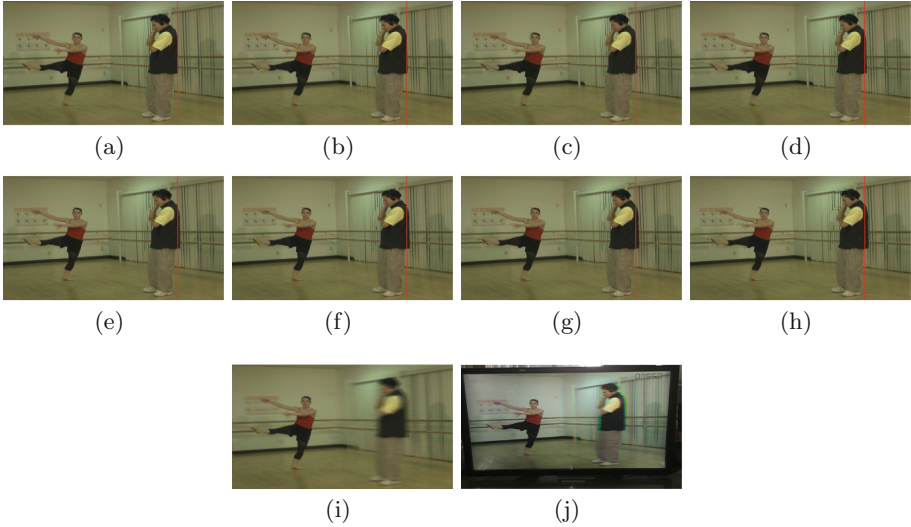
**Table 1.** FPGA utilization summary\*

Resource utilization	Symmetrical shift-sensor		Asymmetrical shift-sensor		Saved
	Used	Utilization	Used	Utilization	
Slice registers	13424	3%	11625	2%	1799
Slice LUTs	12690	6%	11164	5%	1526
Fully used LUT-FF pairs	8441	47%	7461	42%	980
Bonded IOBs	283	47%	252	42%	31
Block RAM/FIFO	121	16%	113	15%	8
BUFG/BUFCTRLs	3	9%	3	9%	0

\*Does not include the recurse consumption of the image fusion module.

Figure 10 shows the original view and the seven virtual views generated by the multiview rendering module. The viewing angles (viewpoints) of those views are slightly different from each other, hence the contents are slightly different. The difference is marked with a red vertical line in the same position of each view in Fig. 10. Go through the red vertical lines from Fig. 10(a) to (h) we can see that the foreground object (man) shifted gradually towards the right side, which indicates that the multiview rendering module can generate correct virtual views. Figure 10(i) shows the fused glasses-free 3D image, which has a good stereoscopic effect when displayed on a 4K glasses-free 3D display (Fig. 10(j)).

As for view fusion, a real-time system that supports view fusion functionality was presented in [9]. The system in [9] supports 1080p@60 Hz and 4k@30 Hz for 4/6/8/16-viewpoint lenticular-based 3DTV displays. Comparing the design in [9], our design implements a lightweight view fusion module that only supports



**Fig. 10.** Eight different views and the glasses-free 3D image fused from them. (a) Reference image  $I_r$ ; From (b) to (h):  $I_1-I_7$ ; (i) Glasses-free 3D image; (j) Stereo effect when (i) was displayed in a 4K glasses-free 3D display.

8-viewpoint fusion for 4k display, hence hardware cost is reduced significantly. In addition, the hardware cost is further reduced since some floating-point multiply operations are avoided in the sub-pixel rearrangement coefficients calculation in comparison with the method in [9].

## 5 Conclusion

This paper presented an FPGA architecture for real-time ultra-high definition glasses-free 3D system. The main contributions of this study are:

- a hardware architecture that supports the  $V + D$  format. As the  $V + D$  video format is used, the system can generate multiviews that required for glasses-free 3D display using DIBR technique. As a result, only one channel of video and its depth map are needed to transmit for this format, hence it requires lower bandwidth in comparison with the traditional 3D video.
- an asymmetric shift-sensor model. With this camera setup, the multiview rendering module in the system can generate 7 virtual views line by line with a few of Block RAMs. Hence the traditional scheme of using DDR for pipelining is avoided. As a result, the hardware cost, as well as the complexity of the system, is greatly reduced. In addition, as the system with the proposed camera setup can generate glasses-free 3D images, it helps to solve the problem of 3D content shortage.

Our proof-of-concept FPGA implementation supports 4K@30 Hz for real-time UHD glasses-free 3D display. A further extension to support 16-viewpoint (or more) autostereoscopic view synthesis can be easily achieved by duplicating the proposed design.

**Acknowledgments.** This work is supported by the National Natural Science Foundation of China (No. 61201347) and the Chongqing foundation & advanced research project (cstc2016jcyjA0103).

## References

1. Liu, R., Tan, W., Wu, Y., Tan, Y., Li, B., Xie, H., Tai, G., Xu, X.: Deinterlacing of depth-image-based three-dimensional video for a depth-image-based rendering system. *J. Electron. Imaging* **22**(3) (2013)
2. Fehn, C.: Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV. In: 2004 SPIE Stereoscopic Displays and Virtual Reality Systems XI, 19 January–21 January 2004, San Jose, CA, United States, pp. 93–104 (2004)
3. Neil, A.: Autostereoscopic 3D displays. *Computer* **8**, 32–36 (2005)
4. Jin, L., Liang, C., Ying, L., Xie, Y.: Design of spaceborne SAR imaging processing and fast verification based on FPGA. In: IET International Radar Conference 2013, pp. 1–5 (2013)
5. Li, Z., Ye, X.S., Zhang, H., Lu, L., Lu, C., Cheng, L.C.: Real-time 3D video system based on FPGA. In: International Conference on Consumer Electronics, Communications and Networks, pp. 469–472 (2014)
6. Lin, T.-C., Huang, H.-C., Huang, Y.-M.: Preserving depth resolution of synthesized images using parallax-map-based DIBR for 3D-TV. *IEEE Trans. Consum. Electron.* **56**(Compendex), 720–727 (2010)
7. Liu, R., Xie, H., Tai, G., Tan, Y., Guo, R., Luo, W., Xu, X., Liu, J.: Depth adjustment for depth-image-based rendering in 3D TV system. *J. Inf. Comput. Sci.* **8**(16), 4233–4240 (2011)
8. Zhang, L., Tam, W.J.: Stereoscopic image generation based on depth images for 3D TV. *IEEE Trans. Broadcast.* **51**(2), 191–199 (2005)
9. Ren, P., Zhang, X., Bi, H., Sun, H., Zheng, N.: Towards an efficient multiview display processing architecture for 3D TV. *IEEE Trans. Circuits Syst. II Express Briefs* **PP**(99), 1 (2016)
10. Liu, R., Deng, Z., Yi, L., Huang, Z., Cao, D., Xu, M., Jia, R.: Hole-filling based on disparity map and inpainting for depth-image-based rendering. *Int. J. Hybrid Inf. Technol.* **9**(5), 145–164 (2016)
11. Liu, R., Xie, H., Tai, G., Tan, Y.: A DIBR view judgment-based fold-elimination approach. *J. Tongji Univ.* **41**(1), 142–147 (2013)
12. Zhao, W.X., Wang, Q.H., Li, D.H., Tao, Y.H., Wang, F.N.: Sub-pixel arrangement for multi-view autostereoscopic display based on step parallax barrier. *Sichuan Daxue Xuebao* **41**(6), 216–218+236 (2011)
13. Tan, W.: Research on key technologies of glasses-free 3D display. Chongqing University (2014)