Real-Time Panoramic Image Mosaic via Harris Corner Detection on FPGA

Lu Chen, Jing Han, Yi Zhang, and Lian-fa Bai^(✉)

Jiangsu Key Laboratory of Spectral Imaging and Intelligent Sense,
Nanjing University of Science and Technology, Nanjing, China
mrblf@163.com

Abstract. To solve the problems such as low matching precision, high algorithm complexity and poor real-time in real-time panoramic image mosaic, this paper makes full use of the significance and parallelism of Harris corners as well as the invariance of feature describing methods in light intensity changes, translation and rotation, and proposes a real-time panoramic image mosaic algorithm which uses Harris corner detection and is logically implemented on FPGA. According to the basic features of classical algorithms and the properties of FPGA, several modules like feature point extraction, description and matching are optimized based on the logical implementation of FPGA. The optimized system realizes the high-precision matching in real time. The new algorithm collects images of 256×256 pixels by CCD camera on Spartan-6 hardware platform of Xilinx. After going through the algorithm, the mosaic images will finally be output on HD display in the form of DVI. The results show that the new algorithm based on FPGA has high precision, good real-time and robustness.

Keywords: Image mosaic · FPGA · Harris corner · Feature descriptor · Real-time

1 Introduction

Image mosaic has become an increasingly popular research field. It performs space matching of a series of images overlapping with each other in the same scene and presents a panorama containing image sequence information and a wide-angle view through image fusion. Currently, the research on static image mosaic algorithm has matured, but there are very few research and application of video mosaic. The key of video mosaic is the real-time of matching. Tsai Du-ming proposed a rapid cross-correlation algorithm of gray window [1], which effectively improves matching speed. But as it uses the matching method based on grayscale, the matching precision is greatly affected by huge image ratio differences and severe regional image deformation, as well as regions with little information. Lowe proposed the matching using SIFT feature descriptor [2, 3]. When conditions like contrast ratio, light intensity and rotation change, image features can still be extracted accurately. This method is widely used in image matching, but it is complex in computation and quite resource-consuming. Wei Zhiqiang and Huang Shuai found that Harris corners hold higher significance than SIFT feature points and are simple in computation [4, 5], effectively reducing resource consumption.

However, due to the limited software running speed, it takes nearly one second to perform feature point matching, which can not meet real-time demands. Yao Lifan optimized several modules of SIFT algorithm and tried to implement image matching through FPGA [6], but he only presented the logic implementation of FPGA in the main direction, rather than the whole matching process. V. Bonato carried out the implementation of FPGA based on SIFT algorithm [7], but due to high complexity of describing operation, this module was performed on FPGA's soft core. The system didn't realize real-time because of the limited running speed of the soft core.

In order to satisfy the demand for real-time in mosaic system, many designers prefer to use FPGA as it holds parallel structures, a large number of logic arrays and fast signal processing. Harris corner detection, which has good parallelism, can perform real-time detection on FPGA and accurately extract corners when images rotate or grayscale changes [8]. The point feature describing method used in SIFT algorithm can provide the matching with highly multi-faceted information of feature points and improve matching precision. This paper uses Harris corner detection to extract feature points and introduces the point feature describing method to match those points on Spartan-6 hardware platform of Xilinx. With its properties and precision invariable, the algorithm is reasonably optimized to run on hardware, which realizes real-time, and acquires panoramic video mosaic images with high precision and stability.

2 The Video Mosaic Algorithm and Its Optimization

2.1 Overlapping Region Estimate

According to the image-forming principle of cameras, the angular transformation of the camera in the horizontal direction is in proportion to the horizontally moving distance of the video image. The formula is as follows:

$$\frac{\Delta\theta}{\theta} \approx \frac{\Delta X}{X} \quad (1)$$

Where $\Delta\theta$ represents the angular transformation of the camera in the horizontal direction. θ means visual angle, which is generally 12 degrees. Smaller object distance makes bigger visual angle. ΔX means the horizontally moving distance of the video image, and X means the width of the video image. The length of the overlapping region of two sequential images is $X - \Delta X$.

Assuming the horizontally rotating speed of the camera is less than 30 degrees per second, and the frame rate is 25 frames per second, then the angular transformation between two frames will be less than 1.2 degrees. It can be roughly estimated that the horizontally moving distance of the video image is less than 10 % of the image width. Subject to hardware conditions, the highest matching precision is acquired when the size of the overlapping region is 50 % of the image size. Hence this paper only perform matching on the half part of the image region. Compared with the matching on the full image, this matching mode can improve matching precision because of the centralized distribution of the feature points, which can also enhance computing speed effectively and save resources.

2.2 Determination of Significant Feature Points

SIFT algorithm can extract 200–800 feature points from an image of 256×256 . By contrast, Harris operator can extract highly significant and stable corners and the calculation is very simple. Harris corners don't perform very well when scales change, but they are still chosen as the significant feature points of the images as scales have little influence on the matching process in video mosaic. Subsequently SIFT feature vector describing method is used to do describing calculation. The time for feature description is in direct proportion to the number of feature points, which thus must be restricted to ensure the real-time of the algorithm. Here the threshold value of CRF set in Non-Maximum Suppression is used to realize adaptive extraction of feature points. The formula to calculate the threshold value of CRF is as follows:

$$CRF_i = CRF_{\max}/n \quad (2)$$

Where CRF is the value from the response function of the pixels, judging whether the Harris corner. $CRF_{\max}$ represents the maximum value of CRF, changing when scenes change. n is threshold coefficient. If the number of feature points goes beyond the limit, this paper can change the threshold value of CRF by changing the value of n to control the number of feature points.

2.3 Optimization of Feature Point Description

Classical SIFT feature vector describing method performs calculation of Gaussian pyramid images. As Gaussian pyramid is not used in the new algorithm, it is necessary to optimize the classical method. Specific procedures are as follows:

(1) Main direction extraction. Lowe's SIFT performs calculation of Gaussian-filtered images to get the gradient whose module value $m(x,y)$ will be processed by Gaussian distribution, where the parameter σ satisfies the formula $\sigma' = 1.5 \sigma$, and the radius of the window is $3 \times 1.5 \sigma$. But this algorithm compute the histogram of the gradient of the 3×3 region around the feature point of the original image. There is no need to do Gaussian distribution processing to module values, which greatly reduces the calculation amount.

(2) The calculated region of the descriptor. Divide the neighbor region around the feature point into 4×4 subregions, and the scale of each subregion equals the region used in the calculation of the corner's main direction, which is 3×3 . Taking bilinear interpolation and coordinate axis rotation into account, the radius of the calculated region should be set as:

$$r = \frac{3 \times \sqrt{2} \times (d + 1)}{2} \quad (3)$$

Calculated results are rounded to integers. As it is very complex to implement circular neighbor regions in FPGA, this paper choose square neighbor regions to simplify the calculation. And there should be a large number of sampling points, so this paper choose a feature point-centered region with the scale of 21×21 for calculation.

(3) The calculation of the coordinates and module values in the descriptor. Due to the change of the calculated regions in the descriptor, the formula used to calculate coordinates and module values should be altered accordingly. The coordinate values of the sampling points in subregions are as follows:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, x, y \in [-r, r] \quad (4)$$

$$\begin{pmatrix} x'' \\ y'' \end{pmatrix} = \frac{1}{3} \begin{pmatrix} x' \\ y' \end{pmatrix} + \frac{d}{2} \quad (5)$$

Where (x,y) represents pixel coordinate, θ means the main direction of the feature point, (x',y') means the rotated coordinate.

Use a weighted Gaussian ($\sigma = d/2$) to process the module values of the gradients in subregions and do additions according to corresponding directions:

$$w = m(x, y) \exp\left(-\frac{(x'/3)^2 + (y'/3)^2}{2 \times (d/2)^2}\right) \quad (6)$$

$$weight = w \cdot dr^k \cdot (1 - dr)^{1-k} \cdot dc^m \cdot (1 - dc)^{1-m} \cdot do^n \cdot (1 - do)^{1-n} \quad (7)$$

where $m(x,y)$ is the gradient magnitude of the pixel. dr and dc respectively represent the contribution pixels make to neighbor seed points in rows and columns. do represents the contribution pixels make to neighbor directions. k , m and n are set as 0 or 1.

2.4 Determination of the Best Matching Points

After the descriptor matching which uses Euclidean distance, one simple and efficient method is used to choose the best matching point in the proposed algorithm. Assuming n pairs of matching points are got through feature point matching and the coordinate values of the i th pair in I_1 and I_2 are (x_{i1}, y_{i1}) and (x_{i2}, y_{i2}) respectively, then calculate the coordinate offsets between them:

$$\Delta x_i = x_{i2} - x_{i1}; \Delta y_i = y_{i2} - y_{i1} \quad (8)$$

Calculate the number of the matched point pairs with the same Δx and Δy . The Δx and Δy of which the number is the largest are considered as the offsets in direction x and direction y of the two images. If none of the matching points has the same Δx and Δy , then the matching is unsuccessful and the mosaic will not be performed.

3 The Hardware Implementation of the Video Mosaic System

For the real-time video mosaic algorithm based on FPGA in this paper, its logic implementation on hardware is mainly made up of three parts, namely adaptive Harris corner detection, feature description and feature matching. The specific procedures are shown as Fig. 1.

Fig. 1. The block diagram of the hardware implementation of the panoramic video mosaic algorithm

3.1 Significant Feature Point Extraction

This module which makes full use of the design concept of the FPGA pipelining can simultaneously store images and use Harris corner detection to finish the feature point extraction of each image in video streaming, and then store the coordinate values of the feature points. The implementation of Harris corner detection is shown as Fig. 2.

Fig. 2. The hardware implementation of Harris corner extraction

(1) The calculation of directional derivatives. To calculate the gradients of the pixels in direction x and direction y, the gradient convolution template is integrated as:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (9)$$

In the implementation of the 3×3 -pixel window, in order to save resources, this paper replaces FIFO IP with a twoport RAM whose depth is 512 datum for the function of automatic line delay of the images. To realize first-in first-out, just like FIFO, this paper only need to set RAM to the mode read first, which will at least reduce half of the resources. The hardware implementation of the matrix of 3×3 is shown as Fig. 3.

Fig. 3. The hardware implementation of the matrix of 3×3

(2) Distributed Gaussian filter. To realize the Gaussian filter of I_x^2 , I_y^2 and $I_x I_y$, the 7×7 filter template whose scale meets $\sigma = 1.5$ is used in this paper. The implementation of the 7×7 window is similar to that of the 3×3 , but the former one needs three RAMs. With the premise that the effect of corner detection is ensured, the filter coefficients generated by Matlab are quantificat, thus the adopted filter template is as follows:

$$G = \frac{1}{255} \begin{bmatrix} 0 & 1 & 2 & 3 & 2 & 1 & 0 \\ 1 & 3 & 6 & 8 & 6 & 3 & 1 \\ 2 & 6 & 12 & 15 & 12 & 6 & 2 \\ 3 & 8 & 15 & 19 & 15 & 8 & 3 \\ 2 & 6 & 12 & 15 & 12 & 6 & 2 \\ 1 & 3 & 6 & 8 & 6 & 3 & 1 \\ 0 & 1 & 2 & 3 & 2 & 1 & 0 \end{bmatrix} \quad (10)$$

Distribution algorithm is a common algorithm for designing digital filters. In this paper, it is used for 2D image filtering, which greatly improves calculating speed and resource use ratio [9]. This paper can see from the 7×7 Gaussian filtering template that it is symmetrical and there are only 8 different filtering coefficients excluding 0. Add the values with the same coefficients in the image window and filter the eight values. Make lookup tables with the size of 2^8 and store the adding results of the products of the i th place of the eight values and their responding Gaussian template coefficients. Make an eight-bit address of lookup table with the i th place of each figure, then output the i th place of the figure having been processed by Gaussian filter. Do shifting operation to restore the output figures and add them to get the Gaussian filtered values of the 7×7 image window. The parallel structure of the distributed filter is shown as Fig. 4.

3.2 Feature Point Description

In order to make the corners possess distinct characteristics which are useful to image matching, this paper performs quantitative description of the gray features of the corners and the regions around them to generate the corner descriptor, which is the key and difficult point in the hardware implementation. As shown in Fig. 5, this part is mainly made up of the calculation of image gradient amplitude and direction, main direction extraction and descriptor generation.

Fig. 4. The parallel structure of the distributed filtering

Fig. 5. The hardware implementation of the main direction and feature description

(1) Calculate gradient amplitude and direction. To ensure real-time, when extracting Harris corners, this paper uses the results of the derivatives in direction x and direction y , to calculate the gradient amplitudes and directions of image pixels:

$$\begin{cases} m(x, y) = \sqrt{(I(x+1, y) - I(x-1, y))^2 + (I(x, y+1) - I(x, y-1))^2} \\ \theta(x, y) = \tan^{-1} \frac{I(x, y+1) - I(x, y-1)}{I(x+1, y) - I(x-1, y)} \end{cases} \quad (11)$$

As it is not easy to realize rooting and trigonometric function on FPGA, this paper uses Cordic algorithm to carry out the operations and saves the calculated results of the amplitudes and directions in RAM for future use [10]. In the calculation, it is very convenient to carry out the hardware implementation of these complex no-linear functions, in that Cordic only uses shift and addition. To enhance system stability, this paper directly invoke Xilinx's build-in core, Cordic IP, to do the calculation.

(2) Extract the main directions of the corners. Read out the amplitudes and directions of the 3×3 region around the corners from RAM, and use $\theta \times \pi/180$ to judge whether the angles are integers [11]. Adding the corresponding amplitudes of the same integral directions to get the gradient histogram. Then compare all the columns of the histogram in pairs and finally get the corner's main direction θ_0 . As the angle calculated through $\tan^{-1}$ is within the range of $(-\pi/2, \pi/2)$, the number of the intervals, 36, will be reduced to 18 after the direction intervals are mapped, which reduces the comparing time and resource consumption.

(3) Describe corners' features. Many systems perform the whole description in DSP or FPGA's build-in embedded softwares to avoid the large amount of calculation [12], which increases costs and reduces calculating speed. This paper skillfully transfers the calculation that is irrelevant to the images to Matlab, and saves the calculated results in the form of lookup tables into registers. This paper only need to read out the corresponding values in the table to fulfill the description.

According to the definition of descriptor, the calculated amount of the description mainly comes from calculating the added module values of the 128-dimensional descriptor and the addresses where those values are added. This paper calculate the post-rotating coordinate interpolation of the weighted amplitude wincoef to get the module value w . The address addr_desc is made up of post-rotating coordinate (x_desc, y_desc) and the direction of the post-rotating 8-column histogram, ori_desc . It can be known from the above algorithm optimization that the calculated region is a matrix of 21×21 . This paper use x, y, ori and theta0 to calculate and read out the addresses of the lookup table and add the calculated results. In the normalization, to avoid decimal arithmetic and to ensure precision, the added results are magnified 1024 times and then input into dividers to get the descriptor of the corner [13].

Specific procedures of the hardware implementation are shown as Fig. 6.

Fig. 6. The hardware implementation of the description of corners' features

3.3 Feature Point Matching

This module is made up of initial feature point matching and the determination of the best matching point pair. The matching relationship of the feature points is got from feature point matching and the coordinate offsets of the images are calculated through the determination of the best matching point pair.

(1) Initial feature point matching. The Euclidean distance used in Classical SIFT matching involves square and root operation, which is not suitable for the implementation on FPGA. In this paper, the sum of absolute differences is used to replace Euclidean distance:

$$D(L_a, L'_b) = \sum_{i=1}^{128} |l_{ai} - l'_{bi}| \quad (12)$$

Where $D(L_a, L'_b)$ is the sum of absolute differences. l_a and l_b represent the corresponding feature vector.

When the state machine detects the rising edge of signal `desc_all_end` in the initial state, it starts the matching made up of two loop nestings. The main procedure of image matching is performed by a state machine, as shown in Fig. 7.

Fig. 7. The state machine of matching

First successively calculate the sums of absolute differences between the descriptor of corner 1 in the Nth frame and the descriptor of each corner in the N + 1th frame, and then sort the results. If the minimum value is less than 75 % of the sub-minimum value, then the two corners are considered as matching points and save the coordinate values of them into registers; If not, then the two corners are not matching points. Continue the calculation of the matching situations between corner 2 in the Nth frame and each corner in the N + 1th frame, and so on. If all the corners in the Nth frame have been detected, the state machine jumps back to the initial state and waits for the starting signal of the next matching process.

(2) The determination of the best matching point pair. After receiving the finishing signal of the feature point matching, `match_end_flag`, this paper read out the coordinate values of the matching point pairs in the register and calculate the coordinate differences:

$$diff_i = \{y_{i2} - y_{i1}, x_{i2} - x_{i1}\} \quad (13)$$

(1) Sort the results of diff_i . This paper uses a brand new comparison sort algorithm in parallel, which is a sort algorithm using the method of time-for-space [14]. This algorithm needs many defined logic variables, but it can fulfill the sort in several clock cycles. Compare the values of diff_i with each other and get the quantized comparing results (1 for greater-than, 0 for less-than). Add the results of comparing one figure with all the other figures, and the sum is the sorting value of this figure in the sequence. Finally, set this sorting value as the address of RAM and save diff_i into RAM in order. When the sort is finished, set sort_end as 1.

(2) Count the numbers of the different values of diff_i . The value corresponding to the largest number is the coordinate offset of the images. Introduce three variables, num , num_max , and diff_best . num represents the number of the value appearing frequently; num_max and diff_best respectively represent the number of the value appearing most frequently and the corresponding coordinate offset. The procedures are shown in Fig. 8.

Fig. 8. The flowchart of the statistic of the coordinate absolute differences

4 Results and Analysis of the Experiments

The image mosaic algorithm in this paper is implemented on Xilinx's Spartan-6 hardware platform. ISE 13.1 and Verilog are used to write programs. The mosaic images of 256×256 pixels are output in the form of DVI. The experimental results are shown as Fig. 9.

4.1 Accuracy Analysis

Take indoor scenes for example, MATLAB is used to simulate the mosaic process of two sequential static images with the classical Harris matching algorithm, classical SIFT matching algorithm, the new matching algorithm in this paper and its adapted version for hardware platform respectively, as shown in Fig. 10 and Table 1. Table 1 shows the

Fig. 9. The experimental results of the system: (a) Experimental equipment; (b) Panoramic mosaic; (c) Indoor scene mosaic; (d) Outdoor scene mosaic.

simulating data acquired by using different algorithms to mosaic the image in Fig. 10(a). This paper can see from the data that, compared with classical algorithms, the new algorithm can extract feature points of higher significance and holds higher matching precision. And its adapted version for hardware keeps excellent matching effect without reducing matching precision.

Table 1. The simulating data of outdoor scenes acquired by using different algorithms

	Feature Points of the left image	Feature points of the right image	Matching pairs	Mismatching pairs
Classical Harris matching algorithm	91	81	9	1
Classical SIFT matching algorithm	206	135	16	3
The new algorithm	39	41	5	0
Adapted version of the new algorithm for hardware	40	39	2	0

Fig. 10. The matching results of different algorithms: (a) The original images; (b) The mosaic image; (c) The classical Harris matching algorithm; (d) The classical SIFT matching algorithm; (e) The new matching algorithm in this paper; (f) The adapted version for hardware

4.2 Real-Time Analysis

To test the real-time effect of the new algorithm on FPGA, this paper compares the running time of each procedure on hardware platform with that on computer, as shown in Table 2. On FPGA, the module of feature point extraction uses the pixel clock input by the video which is acquired and coded by using SAA7113. The frame rate is 25 frames per second and the pixel clock is 24.576 MHz. The other three modules use FPGA's main clock, which is 100 MHz. From Table 2 we can see the total running time of the mosaic algorithm on FPGA is 10 ms, far less than that on Intel CPU with a dominant frequency of 2.5 GHz. And it takes 40 ms to receive a frame of video image, thus the algorithm can satisfy the demand for real-time mosaic.

4.3 Consumption of FPGA's Resources

Spartan-6 XC6SLX150T abounds with logic and storage resources. As shown in Table 3, the algorithm consumes comparatively few logic resources but a great many resources of on-chip RAM. This is because in the implementation most of the complex operations are

stored in on-chip RAM with the form of lookup tables, which reduces logic resource consumption and enhances the algorithm’s running speed. Meanwhile, the video image data for storage and display is saved in off-chip register DDR3 to reduce the consumption of on-chip RAM’s resources.

Table 2. Comparison of the running time needed respectively on FPGA and Intel CPU

	S6-XC6SLX150T-3(ms)	Intel-Core i5 2.5 GHz (ms)
Significant feature point extraction	2.687	1505
Main direction extraction	0.036	16
Feature point description	3.716	406
Image matching	5.464	218
Total time	10.904	2145

Table 3. FPGA Resource consumption

FPGA resources	Used	Available	Utilization
Slice registers	16134	184304	8 %
Slice LUTs	28712	92152	31 %
Multipliers (DSP48A1 s)	14	180	7 %
RAMs	185	268	69 %
Bonded IOBs	16134	184304	8 %

5 Conclusion

This paper, which explores the implementing process of the image mosaic algorithm on FPGA, makes full use of the invariance of feature describing methods of SIFT in light intensity changes, translation and rotation, and applies the property to image matching based on Harris corners. The new algorithm is also optimized according to the features of the hardware. The performance test results of the application of the image mosaic system to video images of 256×256 show that implementing the algorithm on FPGA is of very good real-time and satisfies the demand for high matching precision of sequential images. But due to the limitation of the hardware, how to realize highly precise matching in conditions like huge image scale changes or severe image distortion still needs further research.

Acknowledgment. This work was financially supported by the National Natural Science Foundation of China (61231014).

References

1. Tsai, D., Lin, C.: Fast normalized cross correlation for defect detection. *J. Pattern Recogn. Lett.* **24**, 2625–2631 (2003)
2. Lowe, D.G.: Object recognition from local scale-invariant features. C. In: *Proceedings of the International Conference on Computer Vision* vol. 9, pp. 1150–1157 (1999)
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *J. Int. J. Comput. Vis.* **60**, 91–110 (2004)
4. Wei, Z., Huang, L., Ji, X.: Research on sequence image matching based on point feature. *J. Image Graph.* **14**, 525–530 (2009)
5. Shuai, H., Ke-wei, W., Ling, S.: An image matching algorithm based on Harris scale invariant feature. *J. J. Hefei Univ. Technol.* **34**, 379–382 (2011)
6. Yao, L., Feng, H., Zhu, Y., et al.: An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher. *J. Int. Conf. Field programmable Technol. (FPT)*, 30–37 (2010)
7. Bonato, V., Marques, E., Constantinides, G.A.: A parallel hardware architecture for scale and rotation invariant feature detection. *J IEEE Trans. Circ. Syst. Video Technol.* **18**, 1703–1712 (2008)
8. Amaricai, A., Gavrilu, C., Boncalo, O.: An FPGA sliding window-based architecture harris corner detector. C. In: *2014 24th International Conference on IEEE Field Programmable Logic and Applications (FPL)*, pp. 1–4 (2014)
9. Li, H., Tian, K.: A design of distributed FIR digital filter based on FPGA. *J. Audio Eng.* **36**, 28–32 (2012)
10. Volder, J.E.: The CORDIC trigonometric computing technique. *J. IRE Trans. Electron. Comput.* **EC-8**, 330–334 (1959)
11. Agrawal, K., Chowdhury, S.R.: FPGA based accelerated orientation calculation in SIFT using luts. C. In: *2013 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, pp. 225–227. IEEE (2013)
12. Xiao, H., He, W., Yuan, K., et al.: Real-time scene recognition on embedded system with SIFT keypoints and a new descriptor. C. In: *2013 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1317–1324. IEEE (2013)
13. Leyva, P., Domenech-Asensi, G., Garrigos, J., et al.: Simplification and hardware implementation of the feature descriptor vector calculation in the SIFT algorithm. C. In: *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–4. IEEE (2014)
14. Shi, T., Jin, C.: The comparison sort algorithm in parallel based on FPGA. *J. Digit. Technol. Appl.* **10**, 126–127 (2013)