Design of median filtering system based on FPGA for large windows

Kang Yang ,Min Wei ,Lei Sun

School of Electrical Engineering and Automation Anhui University
Hefei, China
1907804863@qq.com,weimin_51@163.com,499493908@qq.com

***Abstract*—Aiming at the problem of high denoising performance and fast processing speed in the digital image noise suppression process, based on the conventional median filtering algorithm, a scheme for implementing a 5×5 fast median filter using the merged insertion sorting algorithm is proposed. Reduced the number of times to obtain the median value and improve the image processing speed. The algorithm's hardware design was successfully completed in Altera's Quartus II software development environment. Compared with conventional algorithms, the scheme is simple and easy to operate, fast in calculation speed, and can meet the requirements of real-time performance. It is easy to implement on FPGA and provides eliable technical support for the field of image denoising with high real-time requirements.**

Keywords—Merge insertion sorting; fast median filter; comparison times; FPGA; real-time

I.INTRODUCTION

Median filtering is an important part of digital image processing systems[1]. How to reduce the noise introduced in the process of image acquisition and the disturbance produced by the harsh environment, filtering the captured image is the key problem in the digital image processing, and the application of median filter provides a good solution for this problem. In general, there are mainly two means to increase the speed of image processing: First, to change the image processing algorithm, the algorithm is more simple, but the most time-consuming image low-level processing algorithm has been quite mature, and the complexity of its operation is relatively fixed. Therefore, it is very difficult to change the algorithm while ensuring accuracy; the second is to change the way the algorithm is implemented[2~4]. At present, there are many ways to implement image processing algorithms. FPGA technology has a wide range of applications in real-time image processing because of its numerous advantages[5].

Median filtering is a commonly used filtering method, especially for salt and pepper noise[6]. At the same time, median filtering is a kind of spatial nonlinear filtering technology. Its advantage lies in that it can preserve the edge information of the image and can suppress the Impulse noise without damaging the image details. If a common software method is used, the process requires a lot of simple calculations, which will consume a lot of time and affect the processing speed of the system. The use of the structural features of FPGA parallel operations to complete the median filter, a huge amount of data algorithms, can greatly increase the processing speed, real-time image processing. At present, the 3×3 median filter is more commonly used, and is mainly applied to the infrared tracking and target detection field in a complex background. However, when the SNR of a picture is low and the noise density is high, the effect of the 3×3 median filtering is not very satisfactory, resulting in the weakening of the target intensity[7~8].

Aiming at the problem of large amount of data and high real-time requirements when dealing with noise images, this paper designs a median filtering system based on merge-insertion insertion sorting algorithm. Taking a 5×5 filter window as an example, the median value can be obtained. The number of comparisons has been reduced to 90. Compared with conventional methods, this method reduces the number of comparisons and calculations, saves development costs, and is also easily implemented on FPGAs[9].

II.FAST TWO-DIMENSIONAL MEDIAN FILTER ALGORITHM

Median filtering is based on domain operations and can be operated by defining a field of size N × N (N is an odd number) with its center point sliding over an image. The median filter formula is as follows:

$$g(x,y) = median\{f(x-i,y-j)\}\ (i,j) \in S \qquad (1)$$

In equation (1), $g(x,y)$denotes that the median value-filtered grayscale value at the$f(\mathrm{x},y)$point represents the (x,y) point before the median filter. Grayscale value; S represents the median filter template, S represents the median filter template, S can be a square or a circular area. In order to facilitate FPGA implementation, In order to facilitate FPGA implementation, S may take 3×3 templates and 5×5 templates. may take 3×3 templates and 5×5 templates. In the conventional algorithm, the median value of N × Nis calculated, and $\frac{3}{8}(n^4-1)$comparison operations are performed. For the 5×5 sampling window in this paper, 234 comparisons are required. As the number of comparisons increases, the data processing speed becomes slower and the FPGA resource occupancy rate increases. In order to reduce the number of comparisons and resource consumption of the median filter, a merged insertion sort algorithm proposed by Priyadarshan Kolte and Roger Smith et al. was used to implement a 5×5 fast median filter, and the

number of comparisons was reduced to 90[10]. The calculation process for the 5×5 filter window is shown in Fig. 1 (the ascending direction of the arrow indicates the direction)

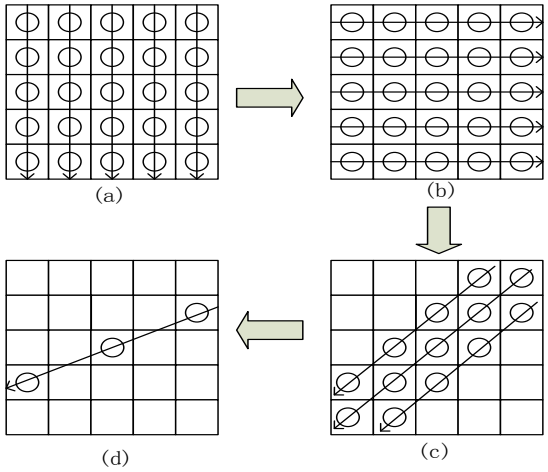


Fig. 1. 5×5 image window for median process

Step1: Sort each column of pixels in the 5x5 filtered window in ascending order according to the arrow direction;

Step2: On the basis of (a), arrange each element in ascending order according to the direction of the arrow;

Step3: On the basis of (b), arrange the 3 elements in the 45-degree diagonal direction in ascending order according to the direction of the arrow;

Step4: Based on (c), the three elements in (d) are arranged in ascending order according to the direction of the arrow. The median value is the median of the entire window.

In the above process, the Merge Insert sort algorithm is used for the sorting of the 5 elements on the row, column, and 45-degree diagonal lines, and the sorting of the 4 elements on the two diagonal lines. The procedure is as follows:

When four elements are compared, taking a, b, c, and d as examples, grouping comparisons a, b, c, and d first is performed. Assume that the obtained results are a>b, c>d, and then the larger of the two results is. Comparing the values, it is assumed that the result is a>c, as shown in Fig.2 (in the ascending order of the arrows).

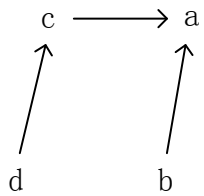


Fig. 2. When a>c>d, the orderly relationship of four elements

At this point, simply determine the location of b and insert it into the ordered sequence of {d,c,a}. Here is the idea of dichotomy, first compare b with c, if b>c, then determine the order of d <c <b <a; If b<c, b, d need to be further compared. Therefore, the worst case of the above 4 numbers only needs to be compared 5 times, and for 4 numbers of bubble sorts, 6 comparisons are needed. For a median value, this does not reflect the higher advantage, but For an image, the number of comparisons saved may be hundreds of thousands or even millions. Similarly, for the five-valued sequence with the most number of uses in this design, take a, b, c, d, e as an example, as shown in Fig. 3:

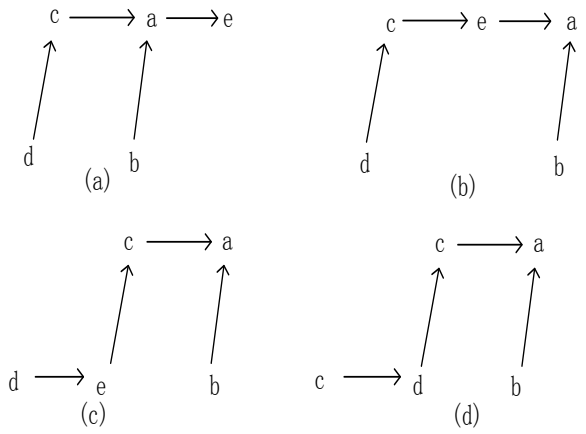


Fig. 3. The orderly relationship between e and b after inserting d, c and a

At the beginning, as with the four elements, the first grouping compares a, b, c, and d, and then compares the larger values of each team, assuming that the first three comparisons are a>c>d and a>b. Get the ordered relationship shown in Fig. 2, then insert e into The proper position in {d, c, a}, using the bifurcation thought only needs to be compared twice, and one of the four situations shown in Fig. 3(a)(b)(c)(d) can be obtained. For any of the above cases, you can always use the idea of dichotomy again to insert b into a suitable location by comparison. For example, as shown in Fig. 3(a), compare b with c to determine whether b is located in the {c, a, e} or {d, c} segments, and if b>c is in the {c, a, e} segment, To get e>a>b>c>d, if b<c, then b, d are required to be compared once to obtain the sorted result. Therefore, the median value can be obtained for the worst case of 5-valued sequence ordering with only 7 comparisons.

Synthesizing the above algorithm can get the median of the sliding window. It needs to sort the 5 numbers 11 times, 4 times 4 times and 3 times at a time. In total, you need to compare $11 \times 7 + 2 \times 5 + 1 \times 3 = 90$times.

III. FPGA HARDWARE IMPLEMENTATION

The system consists of FPGA chip, UART serial port accept module, median filter control algorithm, VGA monitor and SDRAM chip. The UART serial port accept module, the median filter algorithm module, the SDRAM read/write control module, and the VGA display module are implemented in the FPGA chip. The overall structure of the system is shown in Fig. 4:

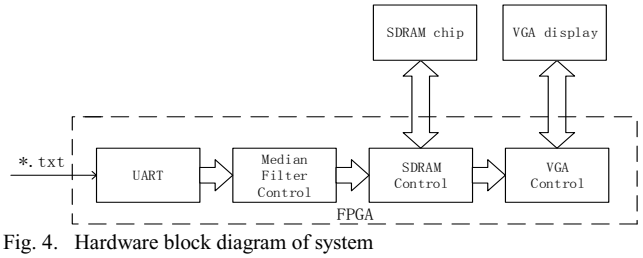


Fig. 4. Hardware block diagram of system

A. SDRAM read and write control module.

Consider the inconsistency of reading and writing timing. If the image data is processed directly, the data will be misplaced. Therefore, this paper uses asynchronous FIFO as the data buffer between FPGA and SDRAM. In order to match the full page operation mode of SDRAM, and make full use of high-speed read and write performance, FIFO is designed to be based on the Pong-Pang operation pipeline structure to implement data cache. At the same time, in order to accept the 8-bit image data after the median filter algorithm and the subsequent supply of VGA display data, the SDRAM is emulated as a double virtual data port (a write port, a read port). The write port is used to receive image data. The read port provides the VGA with previously buffered image data according to the relevant timing signal provided by the VGA, thereby forming a complete frame buffer.

B. VGA display module

The image data is converted into analog signals through the VGA interface circuit, sent to the VGA display to complete the image display, and the data flow control of the image display is realized through the VGA display module. The VGA display module has two main functions: First, the clock signal, line synchronization signal, and field synchronization signal required by the VGA display are generated through programming; Secondly, the pixel data of the SDRAM frame buffer is read out under correct timing control, and at the same time when the display of the current frame is completed, signals are sent to the SDRAM read/write control module, so that the SDRAM read/write module can refresh the data in the frame buffer in time.

C. Sliding window matrix module generation

In order to facilitate the pipeline processing of the algorithm module of the system, it is necessary to ensure that 25 pixel values in the window are output at the same time. In the hardware design of the 5×5 sampling window, 25 registers and 4 shift registers are used in this article. The shift register is With the IP core generated by the FPGA chip, since the image processed in this paper is a 640×480 grayscale image, the bit width of the shift register is set to 8 and the depth is set to 640. Four shift registers are used to buffer the number of pixels in four rows, and registers are used to buffer and output five pixels in each row.

Fig. 5 shows the 5×5 window sampling structure frame. To obtain 25 pixel data in the adjacent 5 rows, you can use the 4 shift registers to first resolve the first 4 rows of storage, and wait until the 5th row of data arrives, and then simultaneously from the 5th row. The data is read in the shift register and the first 4 rows, which ensures that the 5 data in each row are acquired at the same time. Then, the five data in the acquired rows are hierarchically registered. After an appropriate delay, the five data in each row can be taken out at the same time.

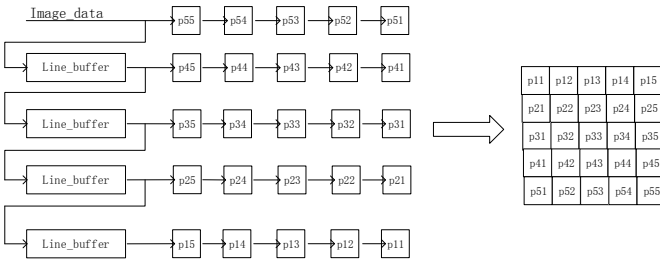


Fig. 5. 5×5 square window structure

Fig.6 shows the 5×5 window simulation waveforms. At the red line of the simulation graph, you can see that each row of the 5 rows of image data can output 5 bits of data and can ensure gradual movement without data loss and misalignment. Therefore, the hardware design of the 5×5 window meets the expected requirements.

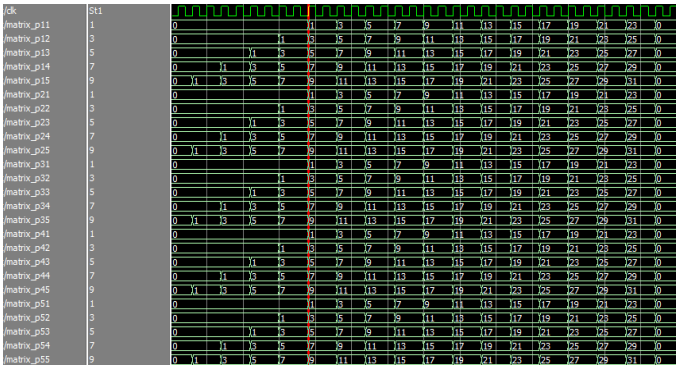


Fig. 6. 5x5 window simulation waveform

D. 5x5 Fast Median Filter Algorithm Module

The purpose of this module is to find the median of 25 numbers and use it as an output. The use of software to implement this algorithm is relatively simple, but usually the implementation of hardware needs to take into account the system hardware resources of the filter. Therefore, from the theoretical analysis, under the premise of achieving engineering application, fewer comparators and registers should be used in the hardware system design process, and the hardware system operation amount should be reduced while obtaining the correct output results. According to the algorithm flow chart of Fig. 1, the Verilog program for calculating the median value is designed. In this program design, 11 times 5 number sorting modules, 2 times 4 number sorting modules, and 1 time 3 number sorting modules are called. The sorting module of 5 numbers needs to be compared at least 7 times, 4 numbers need to be ordered 5 times, and 3 numbers need to be ordered 3 times, for a total of 90 times. 144 times less than conventional comparisons. Fig.7 shows the RTL principle of a 5×5 fast median filter:

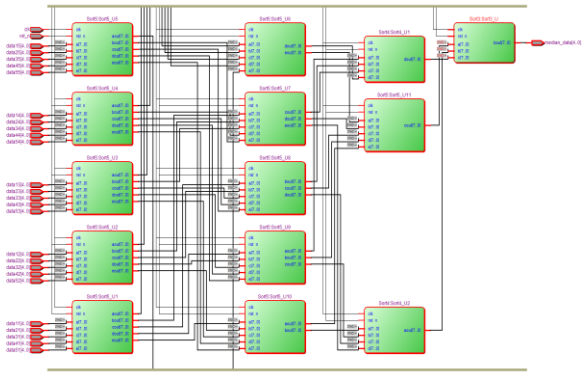


Fig. 7. 5×5 Fast Median Filter RTL Schematic

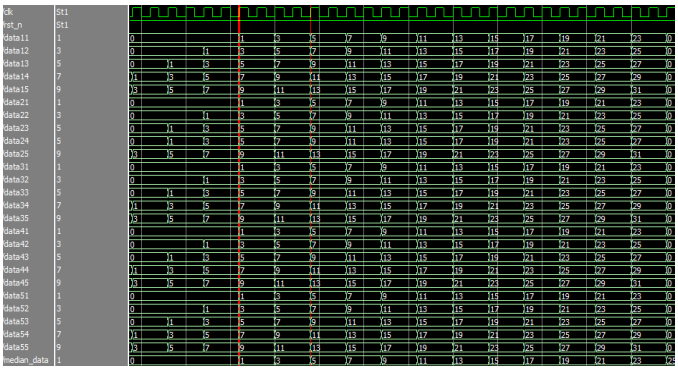
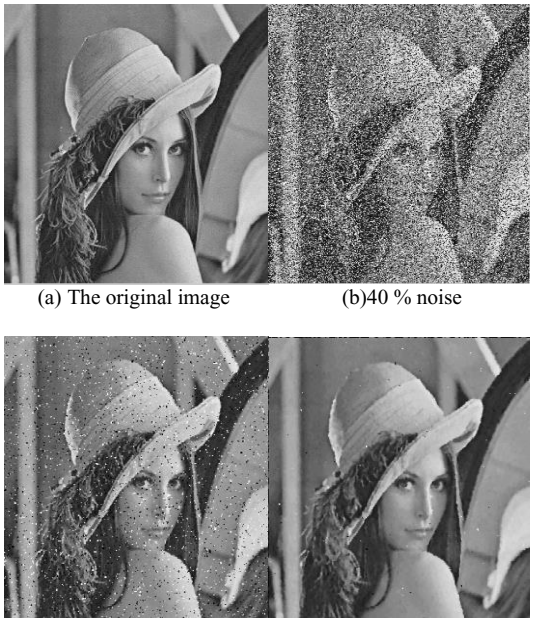


Fig. 8. Median filter algorithm simulation waveform

It can be seen from Fig. 8 that the median of 25 numbers can be output normally in the fourth clock cycle to ensure the correctness and real-time performance of the filtering algorithm. The hardware module design of the algorithm is based on reducing the amount of computation as much as possible. The hardware modules such as row sorting and column sorting that are suitable for implementation on the FPGA are designed using Verilog HDL language, and the design requirements are achieved.

IV. Experimental Results and Analysis

This design uses Altera's Cyclone series EP4CE6F17C8 chip to achieve, and in the Quartus II 13.1 environment using Verilog HDL programming language to complete the circuit design, synthesis, layout, wiring and program download. In order to be able to carry on the emulation test to the system, first use MATLAB to transform a picture into the *.txt format data file, then accepts the module to accept the data through the serial port, after the median value is filtered, sends the data to the SDRAM controller cache, finally passes VGA monitor output. The test result is shown in Fig. 9:



(a) The original image (b)40 % noise

(c) 3×3 window filtering (d) 5×5 window filtering

Fig. 9. Test results

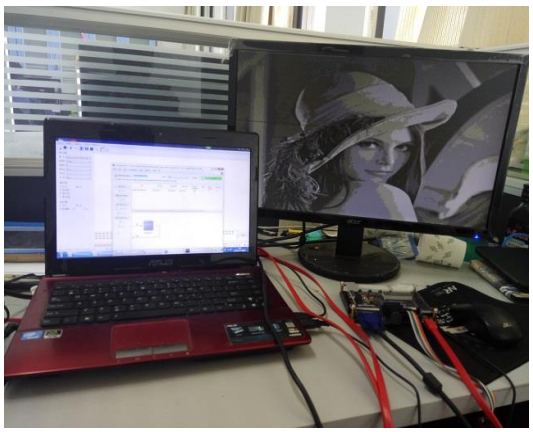


Fig. 10. Hardware set up of experiment

In Fig. 9 (a), (b), (c) and (d) are the comparisons before and after the experimental results. Fig. 9 (a) is the original image; Fig. 9 (b) is to add 40% salt and pepper noise image, Fig. 9 (c) is the image after 3×3 median filter processing, Figure 8 (d) is the design of the 5×5 Fast median filter processed image. The hardware set up of the experiment is shown in the Fig. 10 The hardware set-up is initiated by connecting the JTAG cable from FPGA ALINX AX301 to the laptop or PC as shown in Fig. 10 This ensures loading of sorting algorithm Verilog HDL code into the FPGA.

V. Conclusion

Compared with conventional algorithms, this algorithm can further reduce the number of comparisons of the median filter and improve the speed of operation. In the Quartus II environment, the hardware design of the algorithm meets the

requirements. The algorithm has a simple structure, high real-time performance, less hardware resources, significant denoising effect, and easy implementation on FPGA. It provides a certain technical reference for infrared tracking and target detection in a complex background.

ACKNOWLEDGMENT

This work was supported by the Natural Science Foundation of China(No.61601003),Universities Natural Science Foundation of Anhui Province(No.KJ2013A019, No.KJ2016A836) and the Doctoral Scientific Research Foundation of Anhui University(No.J01001319).

REFERENCES

[1] Rafael C. Gonzalez, and Richard E. woods, Digital Image Processing,3rd edition, Prentice Hall, 2009.

[2] S. Esakkirajan, T. Veerakumar, Adabala N. Subramanyam and C.H.PremChand, "Removal of High Density Salt and Pepper Noise Through Modified Decision Based Unsymmetric Trimmed Median Filter", *IEEE Signal Processing Letters,* Vol. 18, No.5, pp 287-290, May 2011.

[3] Lakshmi Darsi and G. Vara Lakshmi, "Hardware implementation of Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF)" *IOSR Journal of VLSI and Signal Processing,* Vol. 2, No. 6, pp 47-51, Aug2013.

[4] Dr.M.Anand and Narasimha Murthy, "Removal of Salt and Pepper Noise from highly Corrupted Images using Mean Deviation statistical parameter*", International Journal on Computer Science and Engineering (IJCSE)*, Vol. 5, No. 02, pp 113-119, Feb 2013.

[5] H. Hwang and R.A. Haddad, "Adaptive Median Filters: New Algorithms and Results", IEEE Trans. Image Processing, vol. 4, No. 4, pp. 499 - 502, Apr. 1995

[6] S. Zhang and M.A. Karim, "A New Impulse Detector for Switching Median Filter", IEEE Signal Processing Letters, vol. 9, No. 11, pp. 360-363, Nov. 2002.

[7] R.H. Chan, C.W. Ho, and M. Nikolova, "Salt-and-Pepper Noise Removal by Median-Type Noise Detectors and Detail-Preserving Regularization", IEEE Trans. Image Processing, vol. 14, No. 10, pp. 1479-1485, Oct. 2005.

[8] N.I. Petrovic and V. Crnojevic, "Universal Impulse Noise Filter Based on Genetic Programming", IEEE Trans. Image Processing, vol. 17, No. 7, pp. 1109-1120, July 2008.

[9] Y. Dong and S. Xu, A New Directional Weighted Median Filter for Removal of Random-Valued Impulse Noise, IEEE Signal Processing Letters, vol. 14, No. 3, pp.193-196, Mar. 2007.

[10] Priyadarshan Kolte, Roger Smith,When Su,"A fast median dilter using altiVec[C]",IEEE International Conference on Computer Design,pp.384-391.Mar1999.