# LOW COST FPGA IMPLEMENTATION OF ADAPTIVE RANK ORDER FILTER

**Nguyen Van Quan, Nguyen Vu Thang, Vo Le Cuong**

*School of Electronics and Telecommunications*
*Hanoi University of Science and Technology, Vietnam*
*Email: cuong.vole@hust.edu.vn*

Received: XX September 2014, Accepted for publication: YY ZZ 2014

## ABSTRACT

In this paper, two technologies to decrease circuit area of Adaptive Rank Order Filter (AROF) implemented on Field Programmable Gate Array (FPGA) are proposed, namely boundary sorting and unique sorting block methods. The former reduces the number of sorted elements in the biggest-size sorting block from 81 to 32 elements for the largest window size of 9x9. The later uses a unique-maximum-size sorting block for the whole system instead of using 4 separate sorting ones with window sizes from 3x3 to 9x9. The hardware architecture of AROF, which includes the two above methods, along with the common use of window, is implemented on a low-cost Cyclone FPGA of DE2 Altera Development Board. Implementation results show that hardware resource of the whole AROF system with the maximum extended window of 9x9 reduces by 74.33% compared with that of conventional architecture. Among which, 69.01% of the hardware resources is saved by the two proposed methods and the rest 5.32% is brought by the unique window. Meanwhile, the Peak Signal to Noise Ratio (PSNR) and Image Enhancement Factor (IEF) are mostly equal to those of conventional method at different noise densities.

*Keywords.* Adaptive Rank Order Filter, Salt and pepper noise, Boundary sorting method, Unique sorting block method, Low-cost, FPGA

## 1. INTRODUCTION

Impulse noise is very popular in digital image processing. There are some causes which degrade digital image quality by impulse noise such as switching and sensor temperature during image acquisition [1]. The noise has to be removed before an image is processed by other image processing techniques including image segmentation, object recognition, or edge detection. Salt and pepper noise is one type of the impulse noise, which includes minimum and maximum values in gray scale range of an image. There are some research works on removing the noise [2], [3].

The recent fast development of microelectronic allows more powerful hardware for applications of image and video processing. One of the hardware platforms which is popularly used to implement image processing algorithms is FPGA. The method not only contributes to the implementation of complex image processing algorithms at high speed but also makes the design work more flexible. Some researches on FPGA implementation for salt and pepper filter namely Rank Order Filter (ROF), Adaptive Median Filter (AMF), and Adaptive Rank Order Filter (AROF) are discussed in [4], [5]. Among them, AROF has very high filtering efficiency of more than 90% noise density [5]. However, the implementation of this filter requires powerful configurable hardware such as Xilinx Virtex or Altera Stratix.

This paper proposes a new architecture for FPGA implementation of the AROF. In this architecture, two methods are introduced, namely boundary sorting and unique sorting block methods. The first one concentrates on reducing the number of inputs for sorting blocks. The second one uses a unique sorting block instead of separate ones. The hardware architecture of AROF, which includes the two above methods, along with the common use of the single window, is implemented on a low-cost Cyclone FPGA of DE2 Altera Development Board. Implementation results show that the hardware resources of the whole AROF circuit significantly decrease by 74.33% compared to those of conventional one. Meanwhile, the PSNR and IEF stay almost unchanged at different noise densities from 10% to 90%. The method becomes more efficient when the window size increases.

## 2. PROPOSED HARDWARE ARCHITECTURE

In the conventional way of FPGA implementation for AROF, parallel processing technique has been adopted to achieve high speed operation. Fig. 1 shows the conventional architecture [5] which is divided into two main blocks, including *Sliding Window* and *Filter-function*. The conventional architecture requires a lot of hardware resources caused by many separated windows, several sorting blocks and many inputs of each sorting block.

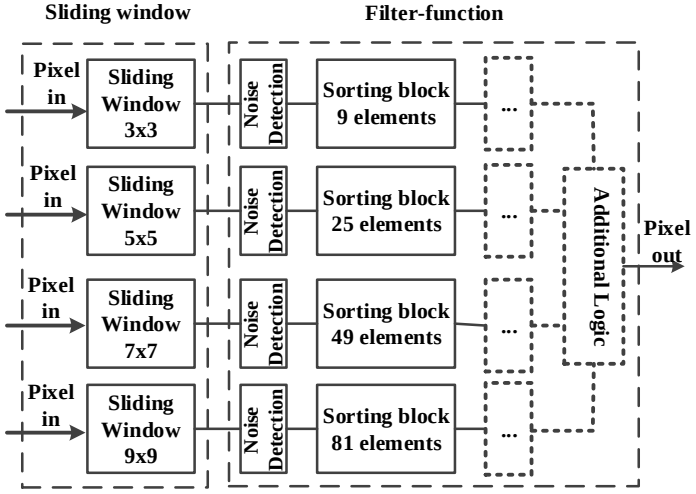


*Figure 1.* Conventional FPGA implementation architecture for AROF

To overcome the drawbacks of the conventional method, this section focuses on a new architecture for FPGA implementation of the AROF. The proposed architecture is based on pipeline and parallel processing techniques to achieve high speed operation as seen in Fig. 2. In the architecture, four different *Sliding Windows*, as in Fig. 1, are combined into one circuit with the maximum-size sliding window of 9x9. The most important change is that two new technologies are integrated into the *Filter-function* block to attain very small circuit area while keeping the high speed, namely boundary sorting and unique sorting methods. The first method reduces the number of inputs for the largest sorting block from 81 in the conventional FPGA implementation architecture to 32 in the proposed architecture. The other utilizes the unique sorting block, *Sorting Block 32 elements*, instead of 4 sorting blocks in conventional architecture, as seen in Fig. 1 and Fig. 2.

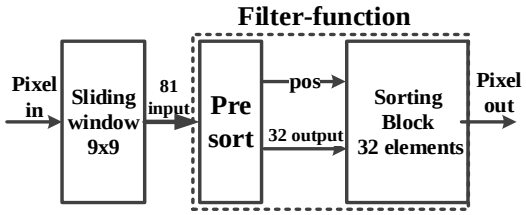


*Figure 2.* Architecture of proposed AROF

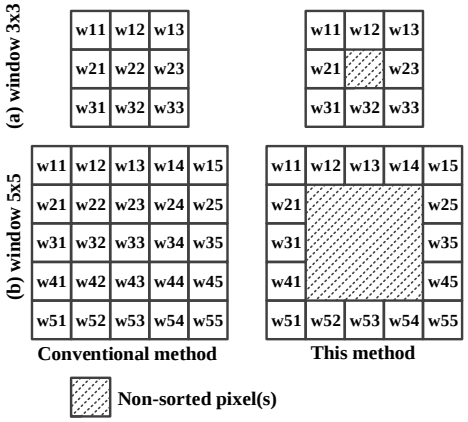


*Figure 3.* Comparison between conventional and proposed methods in terms of number of pixels to be sorted

For the boundary sorting method, when the window size is expanded, the central pixel(s) are not sorted because they are noisy. For example, when the window size is expanded from 1x1 to 3x3, instead of arranging all 9 elements, this method reduces the number to 8 elements surrounding the center pixel as seen in Fig. 3(a). The output is decided based on the 8 sorting values and the noise value of the center pixel. Similarly, when the window size expands from 3x3 to 5x5, instead of sorting 25 pixels, only 16 pixels surrounding the 3x3 window will be sorted, as seen in Fig. 3(b). Table 1 shows the comparison between the proposed and the conventional methods explained in Fig. 2 and Fig. 1, respectively. It is shown that the proposed method is more effective, especially when the window size increases towards 9x9. This helps to decrease the circuit area required for the sorting block of the design. In order to reduce hardware resources more, instead of using multiple sorting blocks, the proposed method uses a unique sorting block for the whole system.

Table 2 shows the comparison between the lists of used sorting blocks by the conventional architecture and those of the proposed architecture which is combined by two mentioned methods. With different extension window size of AROF, from 3x3 to 9x9, the proposed architecture uses the smaller number of sorted pixels than that of the conventional one and a unique *Sorting Block 32 elements*. This sorting block is used instead of *Sorting Block 8, 16 and 24 elements* by adding zero

digits at the beginning of sorted array in order to fill up 32 inputs. The efficiency significantly increases at large extended window size such as 9x9.

*Table 1.* Number of sorted pixels by conventional versus proposed methods

| Window size | Number of sorted pixels | |
|---|---|---|
| | ***Conventional method [5]*** | ***Proposed method*** |
| **3x3** | 9 | 8 |
| **5x5** | 25 | 16 |
| **7x7** | 49 | 24 |
| **9x9** | 81 | 32 |

*Table 2.* List of used sorting blocks by conventional versus proposed architectures

| AROF | Conventional architecture [5] | | Proposed architecture | |
|---|---|---|---|---|
| | ***List of sorting blocks*** | ***Total sorting blocks*** | ***List of sorting block(s)*** | ***Total sorting block(s)*** |
| **AROF 3x3** | 9 | 1 | 8 | 1 |
| **AROF 5x5** | 9, 25 | 2 | 16 | 1 |
| **AROF 7x7** | 9, 25, 49 | 3 | 24 | 1 |
| **AROF 9x9*** | 9, 25, 49 ,81 | 4 | 32 | 1 |

*: AROF with extended window size of 9x9.

### *2.1. Architecture of Sliding window*

Fig. 4 shows an example of 3x3 window architecture [4] in which the data is transmitted through the registers and FIFO blocks serially. Nine registers are required to store 9 pixel values of the sliding window 3x3.

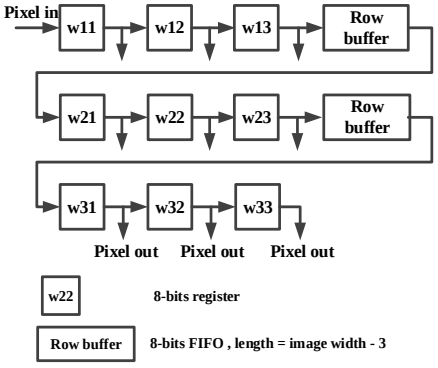


*Figure 4.* Architecture of sliding window

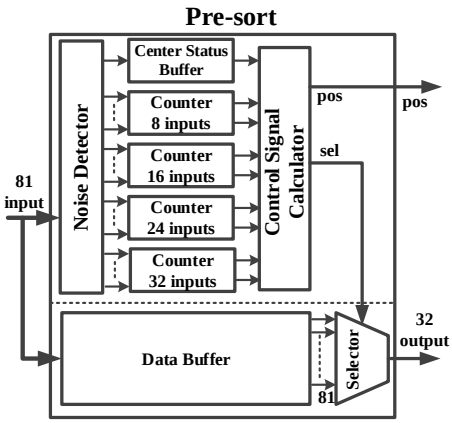


*Figure 5.* Architecture of Pre-sort block

### *2.2. Architecture of Pre-sort*

*Pre-sort* is the most important block which is built to carry the two proposed methods, as seen in Fig. 5. This block receives 81 signals from the *Sliding window 9x9* for calculation then it gives 32 outputs to *Sorting Block 32 elements*. It is divided into two sub-blocks as seen in Fig. 5. While the upper sub-block works as a controller the lower is for data flow purpose.

In Fig. 5, *Noise Detector* detects all noisy pixels transfered from the *Sliding Window 9x9*. Then *Counter 8 inputs* counts the number of noisy pixels from 8 pixels surrounding center pixel as seen in Fig. 3. The block has 2 outputs which export the number of pixels that are equal to 0 and 255 respectively. Similarly, the others are designed and work the same way as the *Counter 8*. After receiving signals from the previous blocks, *Control Signal* block calculates *pos* and *sel* signals to control *Selector* block and *Sorting 32* block, respectively. In parallel, *Selector* block selects 32 outputs from 81 input pixels from *Data Buffer*.

### *2.3. Architecture of Sorting Block 32 elements*

Sorting algorithm used in this paper is Batcher's odd-even merge sort. It is a parallel sorting algorithm and has more efficiency by virtue of a large number of elements sorted, which is suitable for implementation on the hardware. The main components of the sorting network are comparators and registers. A comparator swaps two values; and a register stores temporary data in sorting process [6]. In this paper, all the sorting networks with different number of inputs from 8 to 32 are designed based on the above algorithm.

## 3. IMPLEMENTATION RESULTS AND HARDWARE RESOURCES EVALUATION

### *3.1. Implementation results*

To demonstrate the performance of proposed method, various noise density images are experimented. Fig. 6 shows the filtered and corrupted images from 10% to 90% noise densities. Although several elements are not sorted, the quality of filtered images keeps almost the same as that of conventional method. As seen in Table 3, PSNR and the Image Enhancement Factor (IEF) are similar to those of the conventional structure [5].

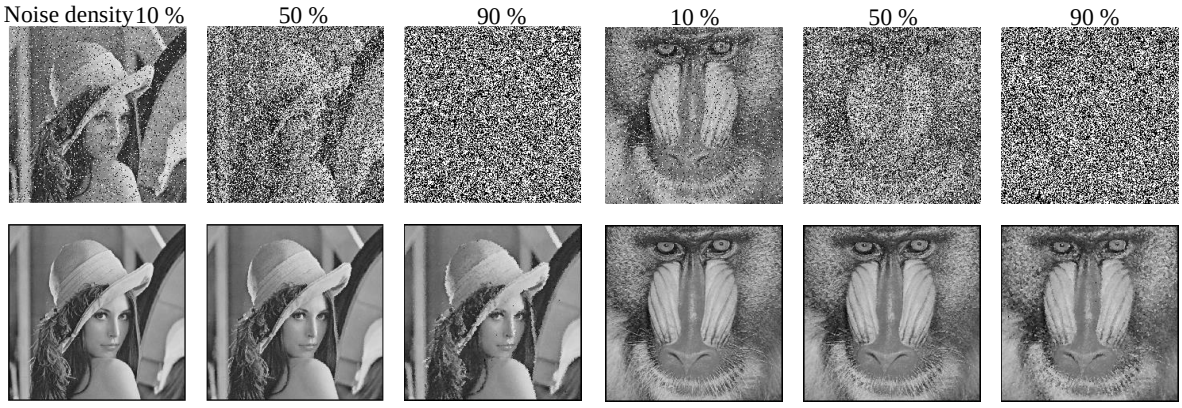


*Figure 6.* 512 x 512 Lena and Bamboo images before and after being filtered

*Table 3.* PSNR and IEF for AROF with proposed hardware architecture vs hardware architecture in [5]

| Noise density | Lena | | | |
|---|---|---|---|---|
| | PSNR | | IEF | |
| | ***Proposed hardware architecture*** | ***Conventional hardware architecture[5]*** | ***Proposed hardware architecture*** | ***Conventional hardware architecture[5]*** |
| 10% | 41.31 | 41.39 | 374.7 | 389.3 |
| 20% | 37.17 | 37.12 | 288.5 | 289.4 |
| 30% | 34.48 | 34.47 | 233.1 | 236.1 |
| 40% | 32.49 | 32.46 | 196.9 | 200.5 |
| 50% | 30.98 | 30.71 | 173.4 | 166.8 |
| 60% | 29.75 | 29.56 | 156.4 | 153.4 |
| 70% | 28.60 | 28.46 | 140.1 | 139.2 |
| 80% | 27.28 | 27.21 | 118.2 | 119.1 |
| 90% | 25.17 | 25.37 | 82.1 | 88.0 |

### 3.2. Hardware resources evaluation

Fig. 7 shows the comparison between hardware resources of the sorting blocks in [5] and our implementation results at the same conditions including sorting algorithm, ISE software for synthesis (ISE 9.2i) and target FPGA (Virtex II Pro, XC2VP50-7ff1152).

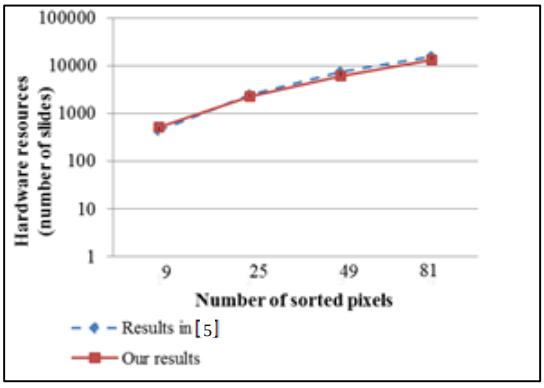


Figure 7. Comparision of results in [5] with our results

The above designs for sorting blocks are implemented on a low-cost Altera FPGA known as Cyclone II on DE2 Altera Development Board. To compare implementation results with those of the original work in [5], the authors follow the following conditions.

(1) Architecture of *Sliding Window* is referenced in [4] , the same references as the original work.

(2) Because [5] does not show the detail design of the filter-function block, the hardware resources of the filter-function block of the conventional work are considered as the total hardware resources of sorting networks.

(3) Hardware resources of the conventional work are calculated based on conditions in (1), (2) by adding the *Sliding Window* and *Filter-function* resources.

*Table 4.* Hardware resources of conventional design versus proposed design

| Device: DE2 Altera Development Board. Available LEs: 33216 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **AROF** | ***Total Logic Elements*** | | | | | | ***Max. Frequency [MHz]*** | |
| | ***Sliding Windows*** | | ***Filter-function*** | | ***Whole system*** | | | |
| | ***CD*** | ***PD*** | ***CD*** | ***PD*** | ***CD*** | ***PD*** | ***CD*** | ***PD*** |
| AROF 3x3 | 245 | 245 | 623 | 746 | 868 | 991 | 235 | 235 |
| AROF 5x5 | 850 | 605 | 4171 | 2121 | 5121 | 2726 | 235 | 235 |
| AROF 7x7 | 1953 | 1103 | 12188 | 4784 | 14141 | 5887 | 235 | 230 |
| AROF 9x9* | 3632 | 1679 | 33074 | 7742 | 36706 | 9421 | 235 | 228 |

CD: Conventional Design in [5], PD: Proposed Design, LEs: Logic elements

* : AROF with extended window size of 9x9

Table 4 shows the comparison of the hardware resources of two main blocks including Sliding window, Filter-function and of the whole system between the conventional architecture in [5] and the proposed architecture. It is shown that, while both blocks contribute to the reduction of hardware resources in the proposed method, the main effect is due to boundary sorting method implemented in the Filter-function block.

*Table 5.* Hardware reduction percentage of proposed architecture compared with conventional one

| | ***Hardware reduction percentage*** | | |
|---|---|---|---|
| **AROF** | ***Sliding Window*** | ***By two proposed methods*** | ***Total*** |
| AROF3x3 | 0% | -14.17% | -14.17% |
| AROF5x5 | 4.78% | 41.98% | 46.77% |
| AROF7x7 | 6.01% | 52.36% | 58.37% |
| AROF9x9 | 5.32% | 69.01% | 74.33% |

Table 5 shows the efficiency of two proposed methods on the whole system in detail. It can be seen that the efficiency of proposed ones improves when the window size increases. For 9x9 maximum window extension, when the boundary sorting and unique sorting block methods are applied, hardware resources of the whole system decrease by 69.01%. This result is 74.33% if both two proposed methods and unique sliding window method are used.

## 4. CONCLUSION

The paper concentrates on proposing the new hardware architecture for FPGA implementation of the AROF. In the design, two methods are introduced. The first one is to reduce the number of inputs of each sorting network, namely boundary sorting method. The second one allows using a unique sorting block instead of several ones separately. The hardware architecture of AROF 9x9 which uses two above techniques saves 74.33% of hardware resource, compared with the conventional design. Even though, in some conditions, there is slight sacrifice of filter performance and the maximum operating speed, the boundary sorting and unique sorting block methods contribute 69.01% out of 74.33%.

## REFERENCES

1. J. Harikiran, B. Saichandana, and B. Divakar, "Impulse noise removal in digital images," International Journal of Computer Applications (0975 – 8887), vol. 10– No.8, 2010, pp. 39-42.
2. Fangzhen Li, Feng Gao, and Nian Cai, "A new algorithm for removing salt and pepper noise from images," Journal of Information & Computational Science 8: 16, 2011, pp. 3819–3825.
3. K. Nallaperumal, J. Varghese, S. Saudia, R. K. Selvakumar, and S. S. Vinsley, "Selective switching median filter for the removal of salt & pepper impulse noise," in Proc. Int. Conf. Wireless Opt. Commun. Netw, 2006, pp. 1–5.
4. Z. Vasicek and L. Sekanina. "An area-efficient alternative to adaptive median filtering in FPGA," In Proc. of the 17th Conf. on Field Programmable Logic and Applications, IEEE Computer Society, 2007, pp. 1-6.
5. M. C Hanumantharaju, M. Ravishankar, D. R Rameshbabu, and S. B Satish, "A novel FPGA implementation of adaptive rank order filter for image noise simple removal," International Journal of Computer and Electrical Engineering, vol. 4, no. 3, 2012, pp. 418-422.
6. Manoj Kumar and Daniel Hirschberg, "An efficient implementation of Batcher's odd even merge algorithm and its application in parallel sorting schemes," IEEE Transactions on Computers, vol. c-32, no. 3, 1983, pp. 254-264.