FPGA Based Efficient Median Filter Implementation Using Xilinx System Generator

Siddarth Sharma¹, K. Pritamdas²

P.G. Student, Department of Electronics and Communication Engineering, NIT Manipur, Imphal, Manipur, India¹
Assistant Professor, Department of Electronics and Communication Engineering, NIT Manipur, Imphal, Manipur, India²

ABSTRACT: Digital Images are an important medium to convey visual information. However, digital images are often corrupted by noise. In this paper, an efficient implementation scheme for median filter is proposed, which is used to remove impulse noise from images. So, the resultant image of the filter is the image with reduced impulse noise. Impulse noise reduction is done using the application of the median filter to the corrupted image by sorting the pixels using a 3x3 window and selecting the median of the window.

KEYWORDS: Impulse Noise, embedded image processing, Xilinx system generator (XSG), FPGA, Simulink, ASIC.

I. INTRODUCTION

Images are an important way of conveying visual information. Digital images tend to get corrupted by noise due to the image sensors, interference in transmission medium or due to atmospheric disturbances. Impulse Noise is the most common noise that occurs in digital images [1]. The most important property about image is that, they can be treated as stationary, and a sliding window can be used which slides over the entire image, successfully placing all the pixels of the image in it[14]. The function of the sliding window is just placement of the pixels in it’s the function of the filter which processes the pixels in the sliding window.

Embedded Image Processing are the best approach towards ASIC (Application Specific Integrated Circuits) as they pave the way for Hardware which can take the input as image do some processing and give the output back as an image, letting the processing to be done by a powerful microprocessor or reconfigurable hardware like FPGA[7,8].

A brief introduction to impulse noise has been given in Section II, describing how it changes the pixels of the image and the noise model for it. In section III, IV and V, the Xilinx system generator design flow has been discussed along the pre-processing and post-processing tasks done to images so that they can be processed by the System Generator blocks. In section V, the performance evaluation of the proposed method is done, and the results are discussed.

II. IMPULSE NOISE

Corruption of digital images by impulse noise [14] is very common. It is independent as well as uncorrelated to the image pixels and occurs randomly over the image. Unlike Gaussian noise, impulse noise only corrupts a subset of the image pixels, the rest will be uncorrupted. There are two types of impulse noises: salt and pepper noise and random-valued impulse noise. This paper mainly focuses on removal of salt and pepper noise from images. In salt and pepper noise the noisy pixels attain either a high value (grey level 255) or low value (grey level 0), hence this noise is termed as salt and pepper, giving the appearance of black and white spots on the images. If \( \gamma \) is the corruption probability, then the salt noise and the pepper noise will have a corruption probability of \( \gamma / 2 \) each. Using the noise model as described in [5], the salt and pepper noise can be modeled as (1):
\[ x_{(i,j)} = \begin{cases} 
0 \text{ or } 255 \text{ with probability } \gamma \\
o_{(i,j)} \text{ with probability } 1 - \gamma 
\end{cases} \] (1)

where \( x_{(i,j)} \) represents the noisy image pixel, where \( i,j \) are the spatial coordinates, \( o_{(i,j)} \) represents the original image.

**III. XILINX SYSTEM GENERATOR DESIGN FLOW**

![System Generator Design Flow](image)

Fig 1 shows the system generator design flow [11,12]. First, the proposed algorithm is designed using MATLAB Simulink. Then the system generator token invokes the respective IP cores for the target FPGA, in the next step it designs the RTL schematic for the proposed method, and it is ready for implementation in the target hardware.

**IV. IMAGE PRE-PROCESSING**

Before feeding the images to the System Generator block, some pre-processing of the image is required. Images are 2-dimensional in nature, and they are to be converted into 1-dimensional[9], a 3x3 processing window is required to process the images, so a 3x3 window generator is used which imitates a processing window, generating 9 pixels at a time as it scans the entire image, this 9 pixels are fed in a parallel manner. Fig 2 shows the window generator used for scanning the entire image and sending the nine pixels in a one dimensional manner.
The “To Frame “ and “Unbuffer” blocks are used to send the elements of the window one by one for each clock cycle. The pixels are ready to be sent to the system generator blocks.

V. MEDIAN FILTER IMPLEMENTATION IN SYSTEM GENERATOR

The Median of nine pixels can be calculated using the traditional sorting method, which is done by arranging the pixels in ascending or descending order and picking the middle value as the median. Or it can be done by calculating the distance between the pixels using the distance norms as discussed in [13]. The pixel with the minimum distance to all the pixels is the median.

In the proposed method, the median is calculated in a different fashion. It is done be sub-dividing the nine pixels into three parts and with each parts containing three pixels. The minimum, median and maximum is calculated for each part. And again, a maximum from the minimums, a minimum from the maximums and a median from the medians is chosen. And from these three, the median is calculated which will be the final median of the nine pixels. Fig 3 shows the block diagram for calculation of median using the method discussed in [4,10].

Before defining the proposed method in Simulink system generator block [6]. The boundary of the FPGA based design is defined using the “Gateway In” and “Gateway Out” block. The “Gateway In” block converts floating-point data to fixed-point data which is readable by FPGA and the “Gateway Out” block converts fixed-point data back to
floating point or a viewable format by MATLAB [2, 3]. Fig 4 shows the “Gateway In” and “Gateway Out” block used for defining the FPGA based design.

![Gateway In and Gateway Out](image)

**Fig. 4 (a) Gateway In and (b) Gateway Out, used for defining the boundary of the FPGA based design**

After the image pre-processing portion is complete, the pixels are ready for calculation and interaction with the FPGA through the Gateway blocks shown in Fig 4. Now, the actual design of the proposed method for median calculation is shown in Fig 5.

![System Generator block for Median calculation](image)

**Fig.5 System Generator block for Median calculation**

**VI. IMAGE POST PROCESSING**

Since the output of the system will be floating point, it needs to be converted to unsigned integer of 8-bit, because data type of image pixels are unsigned integer taking up 8 bits of data, the output pixel will be an individual pixel, that need to be stored in a buffer equal to the size of the image [9] and then converted back to 2D data using the reshape block as shown in fig 6. The image is now ready for viewing in a video viewer block.

![Image post processing blocks](image)

**Fig 6. Image post processing blocks**
VII. RESULTS AND DISCUSSIONS

After the blocks are designed. The target hardware used for the design is Spartan6xc6slx9-3cs324, the images used for testing the design are Peppers image (256x256), desert image (512x512) and Saturn image (512x512) are shown in Fig 7.

The comparison between the traditional sorting method, distance calculation method and the proposed method is done in Fig 8. and the device utilization summary which summarizes the resources used by the proposed design for median calculation is given in Fig 9. The RTL schematic used for the implementation of the design in the target hardware is also shown in Fig 10.
From the results shown in Fig 7, by visual inspection, we can see that the impulse noise in the images have been removed efficiently using the proposed design. From the comparison and the device utilization summary shown in the Fig.8 and Fig.9 we can conclude that the proposed design utilizes lesser number of resources for calculation of the median and can be a better advocate for efficient removal of impulse noise from images.
VIII. CONCLUSION

From the results of the proposed method and the resources utilized by it, it can be concluded that it can serve as a better candidate for removing impulse noise from images efficiently, and can be further extended towards removal of impulse noise from video frames in the fields of video processing, it will have potential applications in the fields of satellite imaging, bio-medical imaging, digital image sensors etc

REFERENCES