Finite State Machine based VHDL Implementation of a Median Filter

Prannoy Ghosh
Electronics Dept, VIT
Pune, India

Akash Nebhwani
Electronics Dept, VIT
Pune, India

Shraddha Dahane
Electronics Dept, VIT
Pune, India

Pranjali Kolwadkar
Electronics Dept, VIT
Pune, India

ABSTRACT

Digital images are often corrupted by impulsive noise also called as salt and pepper noise [1]. It occurs in the form of sharp black or white pixels within the image. An efficient non-linear filter to reduce such noise is the median filter. The main advantage being the preserving of edges as compared to the mean filter. In larger images like satellite images the median filter algorithm needs larger time for processing. A vhdl implementation of such filter shows drastic reduction in processing time. An attempt is made to implement 3X3 median filter on FPGA, using pipeline design and implement the circuit using the concept of finite state machines.

General Terms

Image processing using vlsi.

Keywords

Impulse noise, Median filter, finite state machine.

1. INTRODUCTION

Image noise is random variation of brightness or color information in images, and is usually an aspect of electronic noise[1]. It can be produced by the sensor and circuitry of a scanner or digital camera. Image noise can also occur in the unavoidable shot noise of an ideal photon detector. Image noise is an undesirable by-product of image capture that adds spurious and extraneous information. Medical Images are very often corrupted by various types of noise including speckle noise, salt and pepper noise etc. This corruption of noise is introduced to the original image during image acquisition and transmission. Thus it becomes imperative to de-noise the images for precise data synthesis.

2. MEDIAN FILTER

2.1 Impulse Noise

"Impulsive" noise is sometimes called salt-and-pepper noise or spike noise[1]. An image containing salt-and-pepper noise will have dark pixels in bright regions and bright pixels in dark regions. This type of noise can be caused by analog-to-digital converter errors, bit errors in transmission, etc. It can be mostly eliminated by using dark frame subtraction and interpolating around dark/bright pixels.

2.2 Impulse Noise Reduction

Impulse noise can be drastically reduced by the help of median filter. The median filter is a nonlinear digital filtering technique, often used to remove impulse noise[1]. Median filter is used widely in digital image processing since it preserves edges while removing noise[2]. This is an added advantage over various other filters such as the mean filter.

2.3 Algorithm Description

The main idea of the median filter is to run through the image signal pixel by pixel, replacing each entry with the median of neighbouring entries. The pattern of neighbours is called the "window", which slides, entry by entry, over the entire image. The window has an odd number of entries, so that the median is simple to define: it is just the middle value after all the entries in the window are sorted numerically[3].

3. PROPOSED METHOD

3.1. Algorithm Implementation Issues

Typically, by far the majority of the computational effort and time is spent on calculating the median of each window[4]. Because the filter must process every entry in the signal, for large images, the efficiency of this median calculation is a critical factor in determining how fast the algorithm can run. It has been observed that the median filter can be implemented efficiently using HDL language[5]. This implementation has shown reduced time complexity. Our method uses the concept of finite state machines to implement the median filter .

3.2 VHDL Implementation

An equivalent code is developed using vhdl language. The image is read as a text file. The text file is array of noise affected image pixels. It is generated using matlab. The vhdl code consists of two primary entities which are the sorting and read/write block as shown below

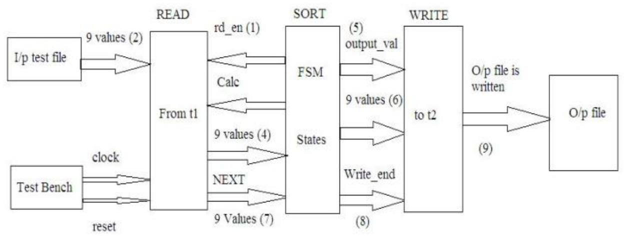

Fig 1: mapping of the two main entities

9 pixel values are read by the reading entity and passed to the sorting entity. After sorting the 9 pixels the middle pixel is replaced by the median value of the 9 pixels. The new values are written to output text file by the write entity. This process continues until the end of image is found. The algorithm is

developed using the concept of finite state machine[4]. The algorithm is explained using FSM concept in the next section.

3.3 Finite States

A finite-state machine is a useful computational model for both hardware and certain types of software. It is considered to be a function that maps states and input to output. The algorithm proposed here works on the concept of finite state machine. The input to the machine is given through the test-bench (reset and synchronous clock). The proposed machine consists of eight states connected sequentially. These states communicate with each other in a sequential manner so as to achieve the desired objective (filtering). These states are Idle, Delay, Delay1, Delay2, Sort, Assign_out, Stop, WR_disable respectively. When reset input is given to the machine, system remains in idle state. On the other hand, it will perform filtering using parallel processing when a synchronous clock is given. It reads and extracts nine values from the input text file and gives it to the Sorting module using Delay, Delay1 and Delay2 states. Sorting is performed in Sort state and the middle value gets replaced in the Assign_out state. Simultaneously, the algorithm reads out the next nine values from the input text file. This parallel implementation reduces the algorithm execution time. Once all the image entries are processed the FSM goes to Stop state where it generates a write enable to the Write block in order to write all the processed values to the output text file. WR_disable disables the write enable to the Write Block and goes back to the IDLE state. The system now waits for the next file input for the whole operation to repeat.

4. SIMULATION AND SYNTHESIS

The algorithm was simulated using Modelsim 10.3 edition and synthesized using Xilinx Spartan 3E.

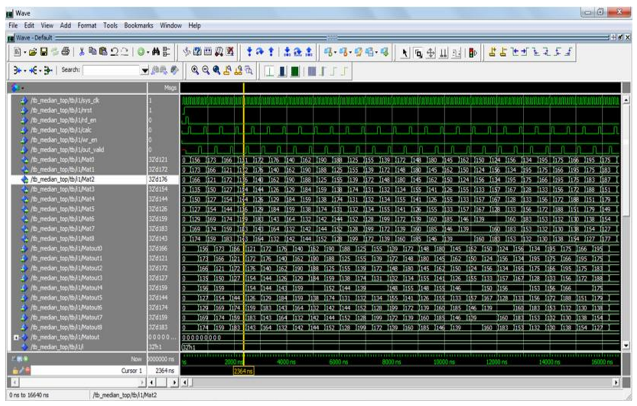

Fig 3: Simulation results

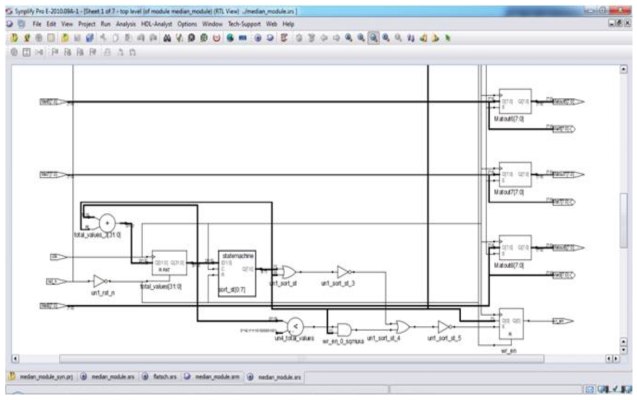

Fig 4: Synthesis results

5. OBSERVATION

Below figure shows the comparison between the noisy image and the denoised image generated using vhdl.

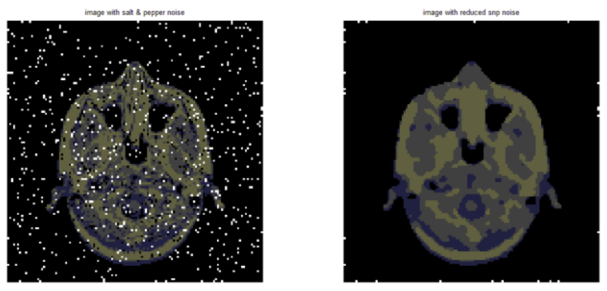

Fig 2: Denoised image

6. COMPARISONS

Processing time was (calculated using the no. of clocks required for processing, with clock freq of 10Mhz) in Vhdl and (Tic-toc function in Matlab).

Parameters	MATLAB	VHDL
Execution time	3.478 sec	34.7 msec
Hardware implementation	Not possible	Possible

7. CONCLUSION

It can be concluded that though the noise is effectively reduced both in MATLAB as well as VHDL, the main advantage of FPGA-based design is the flexibility to exploit the inherently parallel nature of the FPGA for reducing the computational time in the hardware implementation of the median filter algorithm for image processing.

8. FUTURE SCOPE

In future the algorithm can be extended to work on RGB images and portmapping can be done to allocate RAM locations for storing pixel values for less I/O buffer usage.

9. REFERENCES

[1] Jayaraman et al. (2009). *Digital Image Processing*. Tata McGraw Hill Education. p. 272. ISBN 9781259081439.

[2] E. Arias-Castro and D.L. Donoho, "Does median filtering truly preserve edges better than linear filtering?", Annals of Statistics, vol. 37, pp. 1172–2009.

[3] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm", IEEE Trans. Acoust., Speech, Signal Processing, vol. 27, no. 1, pp. 13–18, 1979 Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[4] F. M. Waltz , Ralf Hack "Fast, efficient algorithms for 3x3 ranked filters using finite-state machines", SPIE Conf. on Machine Vision Systems for Inspection VII.

[5] S.Fahmy "Novel fpga-based implementation of median filters for image processing", IEEE conference on Field programmable logic and applications 2005.