Nahrain University, College of Engineering Journal (NUCEJ) Vol.13 No.1, 2010 pp.74-83

Design A Real Time Fast Fuzzy Filter Using Fpga

Assist. Prof. Dr. Khamis A. Zidan
Al-Mustansiriya University
College of Engineering
Computer & Software Eng. Dept
khamis_Zidan@yahoo.com

Assist. Prof.Dr. Dhafer R. Zaghar
Al-Mustansiriya University
College of Engineering
Computer & Software Eng. Dept
drz_raw@yahoo.com

Dr. Ekhlas H. Karam
Al-Mustansiriya University
College of Engineering
Computer & Software Eng. Dept

Abstract

A nonlinear filtering is an important part of processing and restoring image sequence. The computation complexity of these filtering algorithms makes them difficult for real-time processing. This paper presents the design and implementation of a pure fuzzy filter, which is effective to remove the impulse noise, and a new method for hardware implementation into single chip FPGA is proposed. The construction of the fuzzy filter is simple and depends on a set of fuzzy rules to well detect and remove noise pulses. Simulation results show that the fuzzy filter exhibits better filtering properties than standard median filters. The basic FPGA implementation of the filter cannot reach the limit for real time processing with good efficiency. The proposed fuzzy filter is implemented by modifying the input data locations and increase the speed of processing seven times without any increase in the cost.

Key words: Real-time, Fuzzy filter, Xilinx FPGA, Nonlinear filtering, Impulse noise.

1. Introduction

Nonlinear filtering has found many practical applications in signal and image processing. Due to the imperfections of image sensors the images are often corrupted by noise. The impulse noise is the most frequently referred type of noise. In most cases, impulse noise is caused by malfunctioning pixels in camera sensors, faulty memory locations in hardware, or errors in the data transmission [1].

Traditionally, a median filter removes the impulse noise, which is the most popular nonlinear filter. Its hardware implementation is straightforward and does not require many resources [1]. However, the standard median filter [1, 2] has the problem of the edge shift, which means the edge position moves from correct position. Also, it gives a poor performance for images corrupted by impulse noise with higher intensity. A simple median utilizing 3×3 or 5×5-pixel window is sufficient only when the noise intensity is less than 10%. When the intensity of noise is increasing, a simple median filter keeps many shots unfiltered.

Thus more advanced techniques have to be utilized. In recent years, several fuzzy and non-fuzzy techniques for noise reduction have been developed. Delva *et al*. [3] proposed hardware implementation of one of the nonlinear filters, which is based on fuzzy classification of each pixel to subgroups of its neighboring pixels. The criteria of this filter are based on the local contexts, which form the basis of the fuzzy rule. Gupta and Sinha [4] suggest the implementation of the filtering algorithms for smoothing, peak detection and edge detection of fuzzy images using the Xilinx FPGA. The erosion filter forms the core for all of the filtering algorithms and the dilation filter itself is implemented as a function of the erosion filter. Olarte *et al*. [5] presents the architectural proposal and hardware implementation of two fuzzy adaptive filters, which are intended to perform real time non-linear channel equalization. The proposed architecture considers on-line adaptation of Gaussian membership functions.

In this paper, a hardware realization of a pure fuzzy filter is proposed. This filter is based on a set of fuzzy rules in order to detect noise pulses and is able to perform very strong noise cancellation without blurring edges. This filter is implemented as a real time filter using single FPGA chip.

The adopted fuzzy filter is basically a nonlinear mapping of a set of P input variables d_1, d_2, …, d_P to one output variable (Δy). The input variables (d_i) are usually defined as the luminance differences:

$$d_i = x_j - x_0 : \quad x_j \in W \text{ ,j=1,2,…8} \qquad (1)$$

Where x_0 represents the pixel of the image to be processed and $W=\{x_j\}$ is a set of pixels inside

the processing window with the exception of x_0 itself as shown in figure (1). A window of size 3×3 is used to scan across the entire image. Thus, the elements of the window W centered at (i,j) are as follows: $x_1=I_{i-1,j-1}$, $x_2=I_{i-1,j}$, $x_3=I_{i-1,j+1}$, $x_4=I_{i,j+1}$, $x_5=I_{i+1,j+1}$, $x_6=I_{i+1,j}$, $x_7=I_{i+1,j-1}$, $x_8=I_{i,j-1,}$ where I_{ij} is any point in the image matrix.

The output variable (Δy) is the quantity, which is added to x_0 in order to cancel the noise. This filter makes use of two membership functions for two fuzzy sets **pos** (positive) and **neg** (negative). All fuzzy sets are defined in the domain of luminance difference. Note that, the number of membership functions certainly is not necessarily limited to just two. Triangular membership function is used to represent the fuzzy sets. The triangular membership functions are shown in figure (2).

2. Basic Structure of Fuzzy Filter

The nonlinear action of this filter is defined by a set of fuzzy rules in order to detect noise pulses; each rule deals with a particular pattern of neighboring pixels. The structure of the fuzzy filter consists of two sub-rule bases. Each sub-rule deals with one fuzzy set as shown in the following [6]:

First sub-rule

$$\begin{aligned}
R_1&=\min(\mu_1, \mu_2, \mu_3, \mu_4)\\
R_2&=\min(\mu_2, \mu_3, \mu_4, \mu_5)\\
R_3&=\min(\mu_3, \mu_4, \mu_5, \mu_6)\\
R_4&=\min(\mu_4, \mu_5, \mu_6, \mu_7)\\
R_5&=\min(\mu_5, \mu_6, \mu_7, \mu_8)\\
R_6&=\min(\mu_6, \mu_7, \mu_8, \mu_1)\\
R_7&=\min(\mu_7, \mu_8, \mu_1, \mu_2)\\
R_8&=\min(\mu_8, \mu_1, \mu_2, \mu_3)
\end{aligned} \qquad (2)$$

Second sub-rule

$$\begin{aligned}
R_1^* &= \min(\mu_1^*, \mu_2^*, \mu_3^*, \mu_4^*)\\
R_2^* &= \min(\mu_2^*, \mu_3^*, \mu_4^*, \mu_5^*)\\
R_3^* &= \min(\mu_3^*, \mu_4^*, \mu_5^*, \mu_6^*)\\
R_4^* &= \min(\mu_4^*, \mu_5^*, \mu_6^*, \mu_7^*)\\
R_5^* &= \min(\mu_5^*, \mu_6^*, \mu_7^*, \mu_8^*)\\
R_6^* &= \min(\mu_6^*, \mu_7^*, \mu_8^*, \mu_1^*)\\
R_7^* &= \min(\mu_7^*, \mu_8^*, \mu_1^*, \mu_2^*)\\
R_8^* &= \min(\mu_8^*, \mu_1^*, \mu_2^*, \mu_3^*)
\end{aligned} \qquad (3)$$

Where, μ_i represents the membership function for the fuzzy set **pos** ($\mu_{pos}(x_i)$). While, μ_i^* represents the membership function for the fuzzy set **neg** ($\mu_{neg}(x_i)$)

x_1	x_2	x_3
x_8	x_0	x_4
x_7	x_6	x_5

Figure (1) The neighborhood of pixel (x_0)

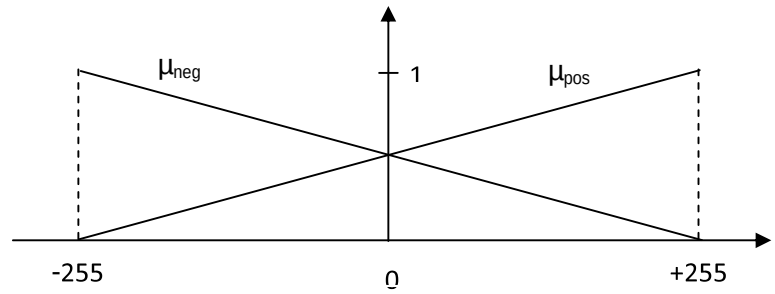


Figure (2) The membership function for the fuzzy set (pos) & (neg).

The overall rule bases of the fuzzy filter are 16 rules. Once a specific rule base has been designed, the output (Δy) is evaluated by means of inference process similar to that found in [6], which can be described by the following groups of relations:

out1= MAX{R_i; i = 1,2,…,8}	4
out2= MAX{ R_i^* ; i = 1,2,….,8}	5
out3= MAX{0(1−(out1+out2)}	6

$$\Delta y = 255 * \frac{out1 - out2}{out1 + out2 + out3}. \quad (7)$$

$$\Delta y' = \Delta y\,(1 - \mu_{small}(|\Delta y|)) \quad (8)$$

Where μ_{small} is a membership function which describes the fuzzy set (small). This function is shown in figure (3).

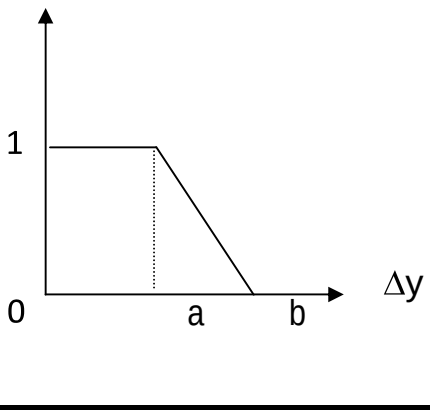


Figure (3) The membership function of fuzzy set (small).

In figure (3), **a** and **b** are tunable parameters. The value of these parameters can be chosen experimentally by analyzing how Mean Square Error (MSE) varies with these parameters.

Equation (8) aims to improve the image edges and fine details by avoiding small luminance correction, which is useless in reducing the effects of noise.

3. Evaluation of Filtering Performance

This section shows the performance of the fuzzy filter. All tests implemented using gold hill image of size 1024×1024 pixels and having 256 gray levels. This image is degraded using impulse (salt and peppers) noise with probabilities 3%, 10%, 13%, 20%, 27%, and 34 %.

In order to have quantitative measure, Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) were computed as follows:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (I_{i,j} - I'_{i,j}) \quad (9)$$

$$PSNR = 10 \log_{10} \frac{(255)^2}{\frac{1}{M \times N} \sum\sum (I_{i,j} - I'_{i,j})^2} dE \quad (10)$$

Where M×N is the number of pixels in image, $I_{i,j}$ is the pixel value in the original image, and $I'_{i,j}$ is the pixel value in the processed image. For the comparison purposes, the results obtained from the fuzzy filter were compared with that obtained from median filter acting on windows of size 3×3 and 5×5. The optimum values of tunable parameters for the fuzzy filter are a=20, and b=35 [6].

Tables (1) and (2), shows the values of MSE and PSNR, respectively, for all filters. From these tables, it can easily be seen that the fuzzy filter has superior performance than median filters for different noise probabilities: the MSE value for the fuzzy filter is several times smaller, while PSNR for this filter is larger than median filter.

It should be mentioned that for higher noise probabilities, the performance of the fuzzy filter could be further improved by implementing the filter iteratively. For example, the value of MSE of the fuzzy filter, which yielded by passing it two times for the noise probability (20%) is (117).

For subjective point of view, figure (4) shows the results of processing (gold hill) image degraded by noise with probability (20%). As it can be easily seen, the median filter with (5×5) window removes noise but blurring the edge and image details, that will showed as additional

noise in the low noise level (less than 30%) as in table (1). The median filter with (3×3) window cannot remove all the noise efficiently. The fuzzy filter can effectively remove all the impulse noise while preserving the image edges and fine details well.

4. Implementation Principles For The System

The main problem in the implementation of this system is the large values of data that fill in the range from 0 to 255. The first step to eliminate this problem in the hardware implementation is the normalization (divided all values in the system and equations by 256) of the input data and as a consequence the normalization of all data in the system. Hence, the input data in equation (1) will convert to the range (0,1) while the values of d_i fill in the range (-1,1). However all the data of a filter equations are in the range (-1,1) and represented as 8-bit for data and 1-bit for sign.

Figure (5) shows the block diagram of U1 that satisfy the positive function in figure (2) after the normalization. This unit is designed to satisfy equation (11). It has four symmetrical and parallel sub-units U11 aligned with to gather. U11 is a sub-unit that has 5 inputs and 3 outputs that is designed to satisfy equations (12 to 15).

$$\mu_{pos} = \frac{1}{2}\left(1' + I_i' - I_o'\right) \quad (11)$$

Where $1' = 255/256 \approx 1$, I_i', I_0' are normalized input data (x_i and x_0 in equation (1)) and μ_{pos} is the membership function for the fuzzy set (pos).

s_1= Boolean sum of (1, a_1, b_1, c_o)	12
c_x= Boolean carry of (1, a_1, b_1, c_o)	13
s_2= Boolean sum of (1, a_2, b_2, c_x)	14
c_1= Boolean carry of (1, a_2, b_2, c_x)	15

Where c_x is internally variable in the sub-unit U11.

The sub-unit U11 requires 5 cells (except for the first unit of U11) and as a result of U1 requires 16 cells with a maximum delay of 4 cell delays.

The second unit is U2 that satisfy equations (4 & 5). The basic blocks for this unit are the sub-unit U21 and U22. U21 is a classic 8-bit comparator used to satisfy equations (16 &17). This sub-unit has 2x8-bit input vector that comes from U1 and 2x8-bit output vectors that represent the minimum and the maximum of the two input vectors respectively, U21 use equation (17) as modification to equation (3) that will reduce the number of inputs and the cost. The function of U22 is the same output as that of U21 except that f2 in U22 will be replaced by equation (18).

f1=min (μ_i, μ_{i+1})	16
f2=max (μ_i, μ_{i+1})	17

Note: max (μ_i, μ_{i+1})= min (μ_{i^*}, μ_{i+1^*})

f3=complement (max (μ_i, μ_{i+1}))	18

The sub-unit U2 in figure (6) require 8 blocks of U21 with 8 blocks of U22 to satisfy the equations (2) & (3).

The sub-unit U21 (U22) requires 30 cells with maximum delay of 3-cell delay. As a result the unit U2 requires 8 U21 and 8 U22 blocks with a total cost of 540 cells and a general maximum delay of 6 cell

Table (1) MSE values for different filters as a function of noise probability.

Noise probability	3%	10%	13%	20%	27%	34%
Noisy	559	1882	2457	3753	5128	6413
Median (3×3)	109	141	146	212	329	620
Median (5×5)	184	233	240	299	338	386
Fuzzy filter	42	74	80	117	157	221

Table (2) PSNR for different filters as a function of noise probability

Noise probability	3%	10%	13%	20%	27%	34%
Noisy	47	35	32	28	25	23
Median (3×3)	63	61	60	57	53	46
Median (5×5)	58	56	55	53	52	51
Fuzzy filter	73	67	66	62	60	56

5. System Implementation

The two units U1 and U2 with their sub-units are the basic blocks that are required to implement the system as shown in figure (7). This system has five pipeline blocks that satisfy the total function of the filter.

The first column in figure (7) contains eight parallel U1 units that satisfy figure (1) for all input mask points. The second column is the unit U2, while the third column is two symmetrical units of U3A that satisfy equation (4) and U3B that satisfy equation (5). U3A is a seven sub-unit of U21 connected as a binary tree as shown in Fig (8), while U3B has same architecture with different inputs and output.

The fourth column (U4) is satisfy the normalized form of equation (7) as in equation (19). This equation is a practical result of combining equations (7) & (8).

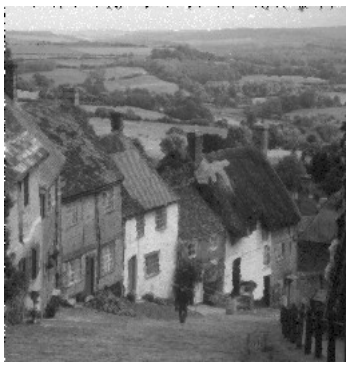
$\Delta y = 255*(out1-out2)$	19

(a)

(b)

(c)

(d)



(e)

Figure (4) Shows the results of processing (gold hill) image corrupted by impulse noise with probability (20%). (a) Original image, (b) Corrupted image, (c) Processed by median filter (5×5), (d) Processed by median filter (3×3), (e) Processed by fuzzy filter.

Equation (8) is implemented by unit U5 as a direct 8x8 LUT function. The details, total cost and speeds is summarized in table (3).

The practical implementation for this system using the chip xcv400-bg432-6 from Xilinx FPGA family shows that this system requires less than 1250 cell and the maximum delay is found less than 10 nsec with eight clock of latency.

6. Efficiency Of Proposed Filter Models

The main aim of this paper is to produce a real time filter using FPGA implementation; deals with a standard 1024x1024 pixel images.

However, to satisfy this aim there is two main models of filter implementation.

First model is the asynchronous model that has memory in the output of filter. This filter model requires (as shown in table (3)) about 73 nsec for output of each point using xcv400 FPGA. That is mean the 1024 x 1024 pixel image will require 77 msec. which is not accepted in real time consideration (up to 20 mesc).

The second model is a high-speed model; that use the pipeline approach by inserting storage units in the output of each sub-unit. This filter model requires (as shown in table (3)) about 11 nsec to have an output of each point using xcv400-bg432-6 FPGA. That is mean the 1024 x 1024 pixel image will require 11 msec. that fall in the range of real time.

The practical results shows the maximum tolerance for this filter model is less than 7%, that because the equations of output of each point depends on the previous processing points for values in the mask in figure (1).

Table (3) The cost and delay of the filter units

Component	Number of units	Cost/(cell)	Delay/(cell delay)	Latency/(cell delay)
U1	8	8 x 16	4	1
U2	1	540	3	2
U3	2	2 x 210	3	3
U4	1	15	4	1
U5	1	128	1	1
Total system	------	**1231**	**4**	**8**

* Cell delay is depends on the type of FPGA its less than 2.5 nsec for xcv400-bg432-6

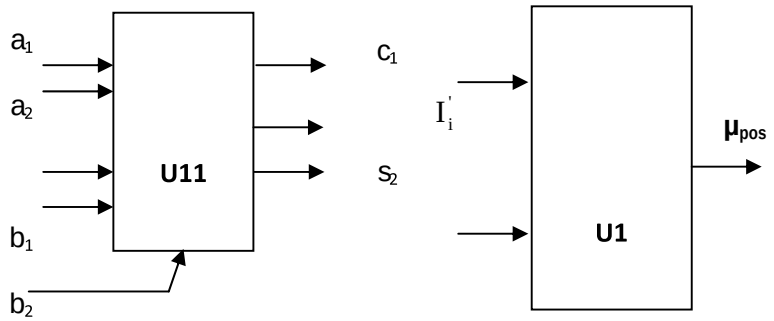


Figure (5) Block diagram of U1 and its sub-unit U11

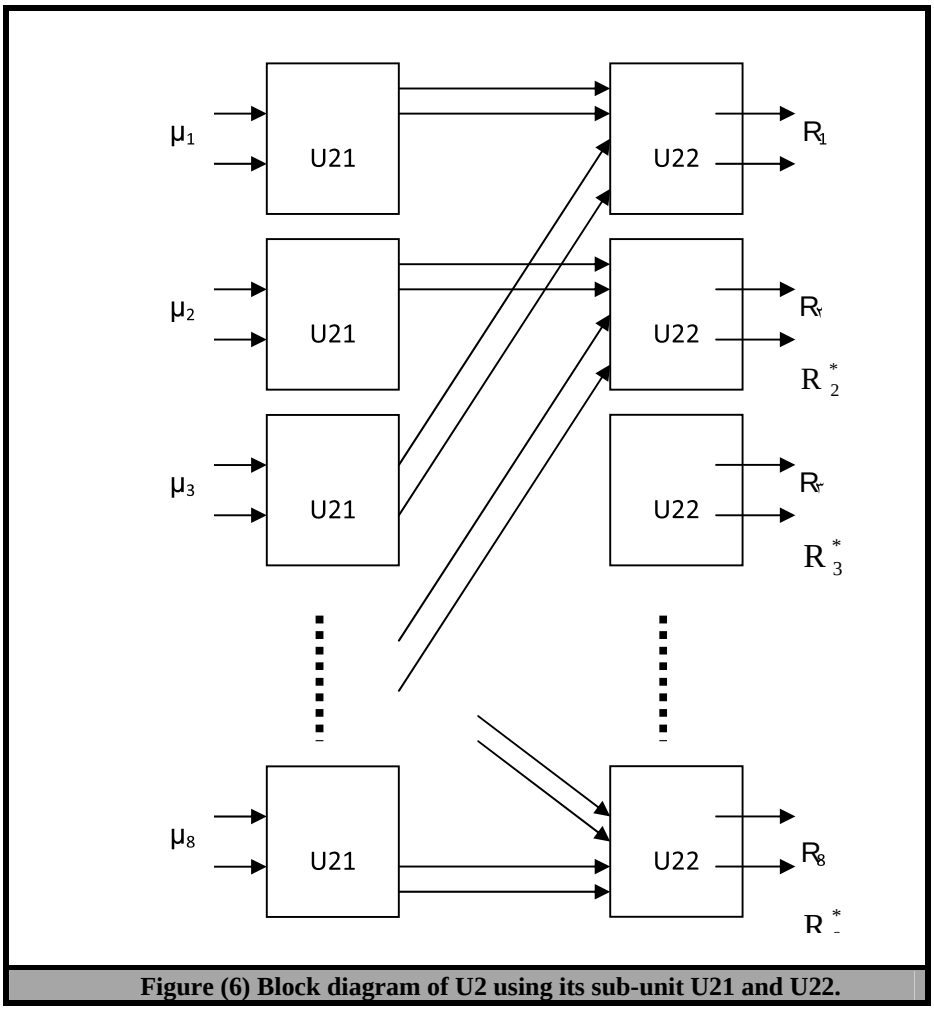


Figure (6) Block diagram of U2 using its sub-unit U21 and U22.

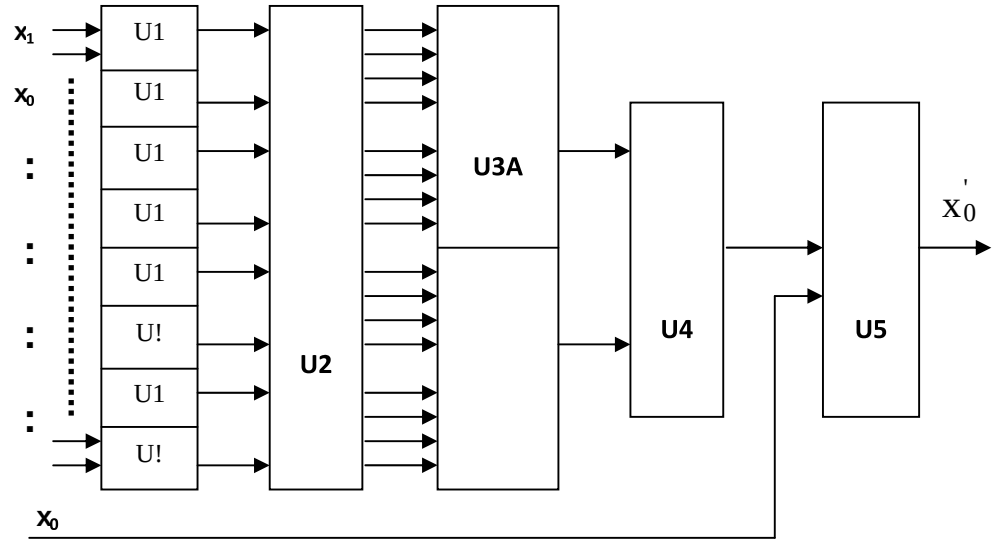


Figure (7) Block diagram of the filter.

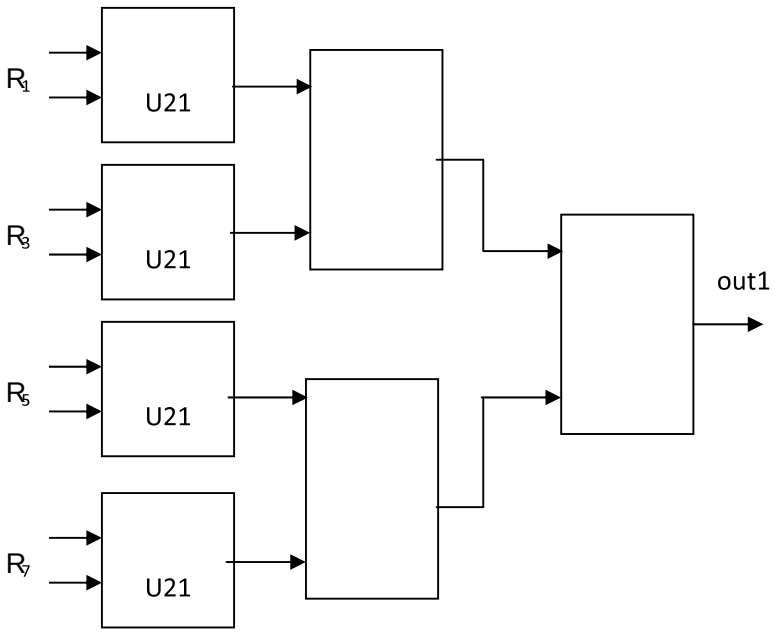


Figure (8) Block diagram of U3A using its sub-unit U21

The proposed model has the advantages of the two above models. The deep analysis for this model shows the fact that the output equation of each point required to processed points for some values in the mask in figure (1) except the first point in each row.

Therefore, the filter should be fed by the first point of the first eight rows serially, and then fed by the second point in the first eight rows serially and so on.

However, in this approach the second point in the row will be fed to the filter after finishing the

process of the first point in the row. In other words this model like implementation of eight synchronous filters with one clock phase shift between each two neighbor filters.

The proposed design requires about 11 msec for 1024x1024 pixel images, which fall in the range of real time. This result is equivalent to the results of model one in tables (1) & (2).

7. Conclusions

In this paper, a new FPGA implementation for the fuzzy filter is proposed. Fuzzy filter exhibits better filtering properties than standard median filters. However, the basic implementation for the filter has low cost (1250 cells) but it execution time is outside real time consideration limit because it require about 77 msec. to be executed.

The proposed method adds a very simple modification to make the filter execution time in the range of real time (11 msec) with the same cost. This modification will not change the design but it will rearrange the input data to use the filter with higher efficiency. The proposed implementation in this paper modifies the input data locations in order to increase the speed to seven times without any increase in the cost.

References

[1] Zdenek V., and Lukas S., "Novel Hardware Implementation of Adaptive Median Filter", Faculty of Information Technology, Brno University of Technology, 2008.

[2] Yoshinori I., Takanori S., and Noritaka Y., "Impulse Noise Detector Using Mathematical Morphology", Graduate School of Science and Technology, Chiba Univ., Japan, 2006.

[3] Delva J. R., Reza A. M., and Turney R. D., "FPGA Implementation of a Nonlinear Two Dimensional Fuzzy Filter", IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, pp. 2143-2146, March 1999.

[4] Gupta N., and Sinha P., "FPGA Implementation of Fuzzy Morphological Filters", SPIE proceedings series, vol. 5, pp. 220-230, 2004.

[5] Olarte F., Ladino P, and Melgarejo M., "Hardware Realization of Fuzzy Adaptive Filters for Non- Linear Channel Equalization", IEEE International Symposium on Circuits and Systems, vol. 2, pp. 932 - 935 , May 2005

[6] Fabrizio R., and Giovanni R., "Adaptive Image Smoothing Using Fuzzy Operators", IEEE Workshop on Nonlinear Signal and Image processing, 1995.

الخلاصة

تعتبر المرشحات اللاخطية من اهم الاجزاء في معالجة الصور، حيث ان تعقيد هذه المرشحات يجعل من الصعب بناءها بتقنية الوقت الحقيقي (real-time). في هذا البحث تم تصميم و بناء مرشح ضبابي (fuzzy filter) لازالة الضوضاء النقطية. و قد تم اقتراح تقنية (FPGA) لبناء هذا المرشح بقطعة واحده. ان تركيبة المرشح تعتمد على مجموعة من القواعد الضبابية المصممة بصورة بسيطة لفحص وازالة الضوضاء النقطية بصورة جيدة. ومن نتائج المحاكاة تبين ان هذا المرشح يعطي نتائج افضل من مرشحات الوسيط التقليدية. وحيث ان البناء الاولي للمرشح لايغطي متطلبات الوقت الحقيقي، لذلك فان الاقتراح الجديد يعتمد على تحوير مواقع الدخل للمرشح مما يؤدي الى زيادة السرعة الى سبعة اضعاف دون اي زيادة في الكلفة