IJRRECS/October 2013/Volume-1/Issue-6/1532-1541 **ISSN 2321-5461**

**INTERNATIONAL JOURNAL OF REVIEWS ON RECENT ELECTRONICS AND COMPUTER SCIENCE**

# AN EFFICIENT REMOVAL OF NOISE USING MDBUT FILTER

**G.Dinesh Kumar[1], P.Vijay Gopal[2]**

**[1]M.Tech, Dept of ECE, Pragati Engineering College, Surampalem, A.P, India**

**Email Id: dinu.409@gmail.com**

**[2]Asst.Prof. Dept of ECE, Pragati Engineering College, Surampalem, A.P, India**

**Email Id: gopal.05432@gmail.com**

## ABSTRACT:

In this paper it's vital to get rid of or minimize the degradations, noises in valuable ancient blurred color pictures. The ancient on the market filtering methodologies area unit applicable for mounted window dimensions solely these area unit not applicable for varied scale pictures. In our project we propose a brand new technique for digital image restoration, in this the noise free and rip-roaring pixels area unit classified based mostly on empirical multiple threshold values. Then the median filtering technique is applied. so noise free pixels are becoming preserved and solely rip-roaring pixels get improved. Here we've written the core processor small blaze is intended in VHDL (VHSIC hardware description language), implemented using XILINX ISE 8.1 style suite the algorithmic rule is written in system C Language and checked in SPARTAN-3 FPGA kit by interfacing a test circuit with the laptop victimisation the RS232 cable. The check results area unit seen to be satisfactory. the world taken and therefore the speed of the algorithm are evaluated.

***Keywords-UART, VHDL, Soft core, Micro blaze, salt& pepper noise, noise removal, median filter.***

## 1. INTRODUCTION:

In this digital world because of the wide usage of communication systems particularly within the method of video transfer from one place to a different like DVSB receivers, digital TV's , Mobile video callings. There area unit possibilities for addition of noise within those videos. Salt and pepper noise is commonest noise gift in those videos. Video could be a cluster of frames running at the same time. So, we tend to changing video into frames at that

time frames into pictures and for that pictures we tend to do the noise removal by exploitation the FPGA.

In image process it's sometimes necessary to perform high degree of noise reduction in a picture before activity higher-level process steps, like edge detection. The median filter could be a non-linear digital filtering technique, typically wont to take away noise from pictures or alternative signals. the concept is to look at a sample of the input and choose if it's representative of the signal. this is often performed employing a window consisting of associate degree odd range of samples. The values within the window area unit sorted into numerical order; the average, the sample within the center of the window, is chosen because the output. The oldest sample is discarded, a replacement sample nonheritable, and also the calculation repeats.

Median filtering is a common step in image processing. It is particularly useful to reduce speckle noise and salt and pepper noise. Its edge-preserving nature makes it useful in cases where edge blurring is undesirableImage synthesis is the process of creating new images from some form of image description. The kinds of images that are typically synthesized include:

- *Test Patterns* - scenes with simple two dimensional geometric shapes.
- *Image Noise* - images containing random pixel values, usually generated from specific parameterized distributions.
- *Computer Graphics* - scenes or images based on geometric shape descriptions. Often the models are three dimensional, but may also be two dimensional.

Synthetic images are often used to verify the correctness of operators by applying them to known images. They are also often used for teaching purposes, as the operator output on such images is generally `clean', whereas noise and uncontrollable pixel distributions in real images make it harder to demonstrate unambiguous results. The images could be binary, grey level or color.

## II NOISE

In common use the word noise means unwanted sound or noise pollution. In electronics noise can refer to the electronic signal corresponding to acoustic noise( in an audio system) or the electronic signal corresponding to the (visual) noise commonly seen as ‘snow’ on a degraded television or video image. In signal processing or computing it can be considered data without meaning; that is, data that is not being used to transmit a signal, but is simply produced as an unwanted by product of other activities. In information theory, however, noise is still considered to be information. In a broader sense, film grain or even advertisements in web pages can be considered noise.

Noise can block, distort, or change the meaning of a message in both human and electronic communication.

In many of these areas, the special case of thermal noise arises, which sets a fundamental lower limit to what can be

measured or signaled and is related to basic physical processes at the molecular level described by well known simple formulae.

**General Concept of Salt and Pepper Noise**

In the following examples, images have been corrupted with various kinds and amounts of drop-out noise. In, pixels have been set to 0 or 255 with probability *p*=1%. In pixel bits were flipped with *p*=3%, and in 5% ofthe pixels (whose locations are chosen at random) are set to the maximum value, producing the snowy appearance.

For this kind of noise, conventional low pass filtering, *e.g.* mean filtering or Gaussian smoothing is relatively unsuccessful because the corrupted pixel value can vary significantly from the original and therefore the mean can be significantly different from the true value. Median filter filter removes drop-out noise more efficiently and at the same time preserves the edges and small details in the image better. Conservative smoothing can be used to obtain a result which preserves a great deal of high frequency detail, but is only effective at reducing low levels of noise.

**III ADAPTIVE MEDIAN FILTER**

Comparing with Standard median filtering the Adaptive median filtering is an advanced method. Which pixels in an image have been affected by impulse noise can be determined by using spatial processing.AMF performs in the image by comparing each pixel with its surrounding neighbor pixels to classify pixels as noise. The neighborhood pixel of the size is adjustable, as well as for the comparison the threshold is adjustable. A pixel is not structurally aligned with those pixels to which it is similar, as well as pixel that is
Different from a majority of its neighbors can be treated as impulse noise. The median pixel value of the pixels in the neighborhood can be replaced in the place of noise pixels that have passed the noise labeling test.

**Level A:**
IF Zminimum < Zmedium < Zmaximum, then Zmedium is not an impulse go to level B to test if Zxy is an impulse
ELSE
Zmedium is an impulse
The size of the window is increased and
Level A is repeated until
Zmedium is not an impulse and go to level B or
Smaximum reached: output is Zxy
**Level B:** IF Zminimum < Zxy < Zmaximum,
Then
Zxy is not an impulse
Output is Zxy (distortion reduced)
ELSE
either Zxy = Zminimum or Zxy = Zmaximum
Output is Zmedium (standard median filter)

Zmedium is not an impulse (from level A)

## IV MDBUT MEDIAN FILTER

MDBUT Median Filter can be called as Modified Decision Based Un-symmetric Trimmed. It processes the corrupted images by first detecting the impulse noise. Whether it is noisy or noisy free can be checked by using the processing pixel. It can say it is noise free pixel if the processing pixel lies between maximum and minimum gray level values otherwise it can say that it is noisy pixel.

In many practical cases of image processing, only a noisy image is available. This circumstance is known as the blind condition. Many denoising methods usually require the exact value of the noise distribution as an essential filter parameter. So, the noise estimation methods in the spatial domain use the variance or standard deviation to estimate the actual added noise distribution. But it is found that the mean deviation provides better results than the variance or standard deviation to estimate the noise distribution. The advantage of this approach is that the mean deviation is actually more efficient than the standard deviation in practical situations [9].The standard deviation emphasizes a larger deviation; squaring the values makes each unit of distance from the mean exponentially (rather than additively) larger [10]. The larger deviation will cause overestimation or underestimation of the noise. So, we assume that use of the mean deviation may contribute to more accurate noise estimation. Keeping these points in view, the authors have used the mean deviation parameter in deciding the noise pixel and replaced the central pixel by its mean deviation instead of its mean. The steps in the proposed algorithm are given below.

**Step1:** Select 2-D window of size 3x3.

**Step2 :** If this pixel value lies between 0 and 255,pixel.so, no processing is required and its value is left unchanged

**Step 3:** If Pij = 0 or 255, it indicates that the pixel is corrupted by salt and pepper noise .Here two cases areconsidered

**Case i:** The selected window contains few 0 or 255 elements and other elements lie between 0 and255.Thenthe 0 and 255 elements are discarded and the median of the remaining elements is found. The Pij pixel is replaced with this median value.

**Case ii:** Suppose the window under consideration has all the elements either 0 or 255 .Then median ofTheseelements may also be either 0 or 255 which is again a noisy element. Now, find the mean deviation or absolute mean deviation of the window which can never be 0 or 255.Replace the pixel Pij with this mean deviation value

**Step 4:** Apply the steps 1 to 3 for all the pixels in the image for complete processing.

***Case (I):*** If the selected window contains salt/pepper noise as processing pixel (i.e., 255/0 pixel value) and neighboring

pixel values contains all pixels that adds salt and pepper noise to the image:
**Where 255 is processing pixel P (i,j)**

| | | |
|---|---|---|
| 0 | 255 | 0 |
| 0 | 255 | 255 |
| 255 | 0 | 255 |

Since all the elements surrounding are 255's & 0's. It will be either 0 or 255 which is again noisy, if one takes the median value. To solve this problem, the processing pixel is replaced by the mean value & the mean of the selected window is found. Here the mean value is 170. Replace the processing pixel by 170.

***Case(ii):*** If the selected window contains salt or pepper noise as processing pixel (i.e., 255/0 pixel value) and neighboring pixel values contains some pixels that adds salt (i.e., 255 pixel value) and pepper noise to the image:
**Where 0 is processing pixel P(i,j)**

| | | |
|---|---|---|
| 78 | 90 | 0 |
| 120 | | 255 |
| 97 | 255 | 73 |

Now eliminate the salt and pepper noise from the selected window. That is, elimination of 0's and 255's. The 1-D array of the above matrix is [78 90 0 120 0 255 97 255 73]. After elimination of 0's and 255's the pixel values in the selected
Window will be [78 90 120 97 73]. Here the median value is 90. Hence replace the processing pixel by 90.

***Case (iii):*** If the selected window contains a noise free pixel as a processing pixel, it does not require further processing. For example, if the processing pixel is 90 then it is noise free pixel:
Since "90" is a noise free pixel it does not
require further processing.

| | | |
|---|---|---|
| 90 | 95 | 90 |
| 45 | 65 | 21 |
| 32 | 90 | 101 |

Each and every pixel of the image is checked for the presence of salt and pepper noise. Different cases are illustrated in this Section. If the processing pixel is noisy and all other pixel values are either 0's or 255's is illustrated in
Case (i).
If the processing pixel is noisy pixel that is 0 or255 is illustrated in Case (ii).
If the processing pixel is not noisy pixel and itsvalue lies between 0 and 255 is illustrated in

Case (iii).

## IV. ILLUSTRATION OF THE PROPOSED ALGORITHM

This section explains the proposed algorithm with a flow chart and numerical examples. In the processing methodology the entire image must be checked for the presence of noise. The flow chart of the algorithm is shown in Fig.1

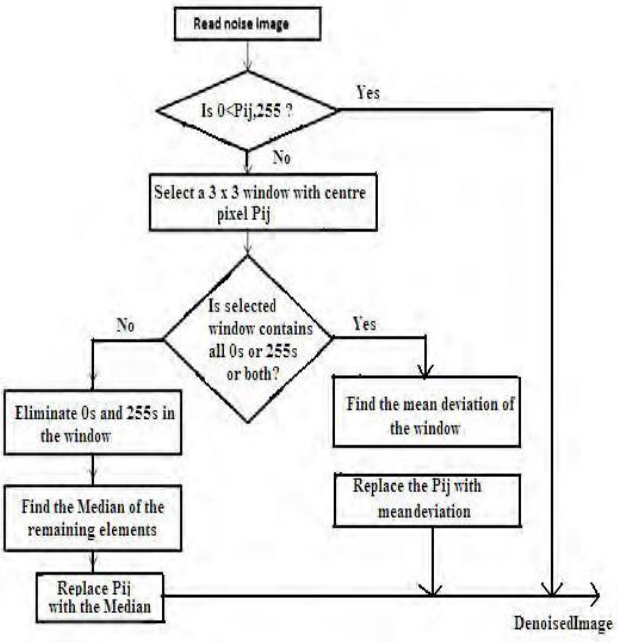


**Figure 1. Flow chart of the proposed algorithm**

## BACKGROUND

The backbone of the architecture is a single-issue, 3-stage pipeline with 32 general-purpose registers (does not have any address registers like the Motorola 68000 Processor), an Arithmetic Logic Unit (ALU), a shift unit, and two levels of interrupt. This basic design can then be configured with more advanced features to tailor to the exact needs of the target embedded application such as: barrel shifter, divider, multiplier, single precision floating-point unit (FPU), instruction and data caches, exception handling, debug logic, Fast Simplex Link (FSL) interfaces and others.

This flexibility allows the user to balance the required performance of the target application against the logic area cost of the soft processor MicroBlaze also supports reset, interrupt, user exception, and break hardware exceptions. For interrupts, MicroBlaze supports only one external interrupt source (connecting to the Interrupt input port) (2). If multiple interrupts are needed, an interrupt controller must be used to handle multiple interrupt requests to MicroBlaze shown in figure l.

An interrupt controller is available for use with the Xilinx Embedded Development Kit (EDK) software tools. The processor will only react to interrupts if the Interrupt Enable (IE) bit in the Machine Status Register (MSR) is set to 1. On an interrupt the instruction in the execution stage will complete, while the instruction in the decode stage is replaced by a branch to the interrupt vector (address Ox 10). The interrupt return address (the PC associated with the instruction in the decode stage at the time of

the interrupt) is automatically loaded into general-purpose register. In addition, the processor also disables future interrupts by clearing the IE bit in the MSR. The IE bit is automatically set again when executing the RTlD instruction. Writing software to control the MicroBlaze processor must be done in C/C++ language. Using C/C++ is the preferred method by most people and is the format that the Xilinx Embedded Development Kit (EDK) software tools accept. The EDK tools have built in C/C++ compilers to generate the necessary machine code for the MicroBlaze processor.

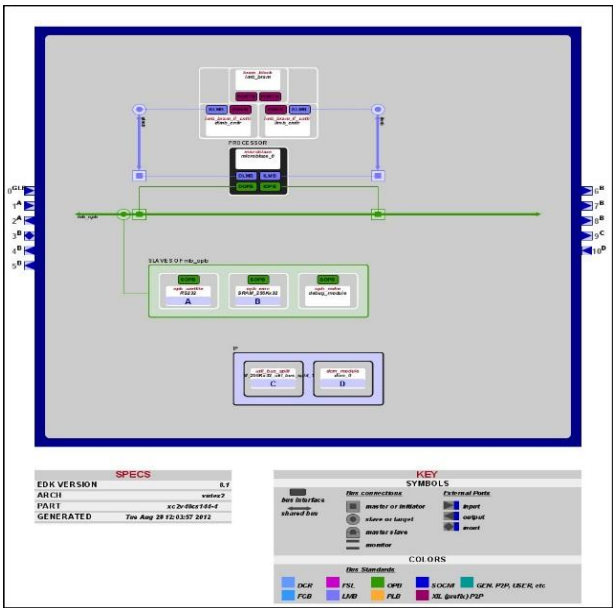


**Figure3: Microblaze Architecture Block Diagram**

Due to the advancement in the fabrication technology and the increase in the density of logic blocks on FPGA, the use of FPGA is not limited anymore to debugging and prototyping digital electronic circuits. Due to the enormous parallelism achievable on FPGA and the increasing density of logic blocks, it is being used now as a replacement to ASIC solutions in a few applications where the time to market is critical and also entire embedded processor systems are implemented on these devices with soft core processors embedded in the system. With the advancement of Field Programmable Gate Arrays (FPGAs), like addition of substantial amounts of memory, a new trend has emerged in the design community to implement the microprocessors on the FPGAs. These kind of processors implemented on a reconfigurable fabric are called soft-processors or soft cores as the design of the microprocessor is available in the form of software bit stream which can be downloaded on FPGA by the user. The users have the choice of selecting the resources on the processor and the memory hierarchy. Soft cores are designed to meet minimum performance specifications over a range of technology implementations, even though core performance varies across technologies. Soft cores are technology independent and require only simulation and timing verification after synthesized to a target technology. This reduces the design cycle development time by a major factor as compared to the development cycle for a hard core processor and has the advantage of customizing the soft core design for a specific application. Currently there are a number of soft cores available in the markets that are developed by giants in the

field of reconfigurable devices like Xilinx. Xilinx has their own architecture named MicroBlaze in this arena and they have also ported the popular PowerPC architecture for use in embedded systems. These soft cores are available in the form of synthesized HDL modules or gate level net lists. System designers can embed these cores into their designs and optionally add peripherals to the core.

## IV EXPERIMENTAL SETUP

### Xilinx Platform Studio

The Xilinx Platform Studio (XPS) is the development environment or GUI used for designing the hardware portion of your embedded processor system. Xilinx Embedded Development Kit (EDK) is an integrated software tool suite for developing embedded systems with Xilinx Micro Blaze and PowerPC CPUs. EDK includes a variety of tools and applications to assist the designer to develop an embedded system right from the hardware creation to final implementation of the system on an FPGA. System design consists of the creation of the hardware and software components of the embedded processor system and the creation of a verification component is optional. A
Typical embedded system design project involves: hardware platform creation, hardware platform verification (simulation), software platform creation, software application creation, and software verification. Base System Builder is the wizard that is used to automatically generate a hardware platform according to the user specifications that is defined by the MHS (Microprocessor Hardware Specification) file. The MHS file defines the system architecture, peripherals and embedded processors]. The Platform Generation tool creates the hardware platform using the MHS file as input. The software plat defined by MSS (Microprocessor Software Specification) file which defines driver and library customization parameters for peripherals, processor customization parameters, standard 110 devices, interrupt handler routines, and other software related routines. The MSS file is an input to the Library Generator tool for customization of drivers, libraries and interrupts handlers.

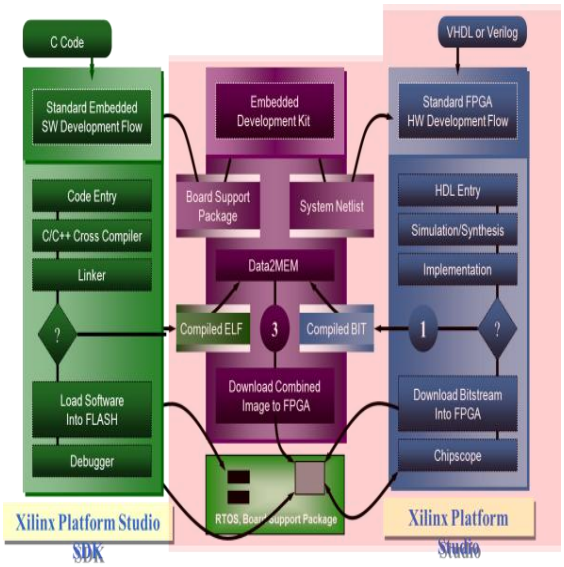


**FIG: Embedded Development Kit Design Flow**

The creation of the verification platform isoptional and is based on the hardplatform. The MHS file is taken as an inputby the Siegen tool to create simulation filesfor a specific simulator. Three types ofsimulation models can be generated by theSiegen tool: behavioral, structural andtiming models. Some other useful toolsavailable in EDK are

Platform Studio whichprovides the GUI for creating the MHS andMSS files. Create / Import IP Wizard which allows the creation of the designer's own peripheral and import them into EDK projects. Platform Generator customizes and generates the processor system in the form of hardware net lists. Library Generator tool configures libraries, device drivers, file systems and interrupt handlers for embedded processor system. Bit stream initializes the instruction memory of processors on the FPGA shown in figure2. GNU Compiler tools are used for compiling and linking application executables for each processor in the system [6]. There are two options available for debugging the application created using EDK namely: Xilinx Microprocessor Debug (XMD) for debugging the application software using a Microprocessor Debug Module (MDM) in the embedded processor system, and Software Debugger that invokes the software debugger corresponding to the compiler being used for the processor. C. Software Development Kit Xilinx Platform Studio Software Development Kit (SDK) is an integrated development environment, complimentary to XPS, that is used for C/C++ embedded software application creation and verification. SDK is built on the Eclipse opensource framework. Soft Development Kit (SDK) is a suite of tools that enables you to design a software application for selected Soft IP Cores in the Xilinx Embedded Development Kit (EDK).The software application can be written in a "C or C++" then the complete embedded processor system for user application will be completed, else debug & download the bit file into FPGA. Then FPGA behaves like processor implemented On Xilinx Field Programmable Gate Array (FPGA) device.

## V RESULTS

The Algorithm is implemented in Microblaze Processor and the results are furnished in the tabulation below

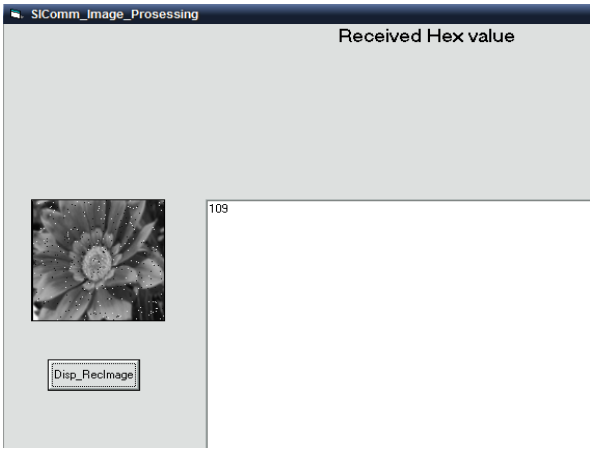


**Fig: Noise Image reading through VB**

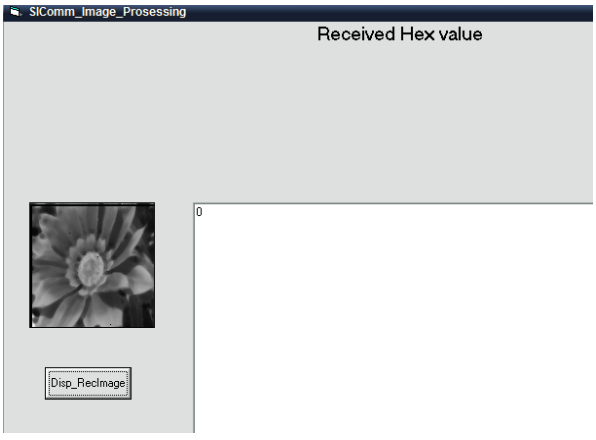


**Fig: Noise removed image**

```
Device utilization summary:
---------------------------

Selected Device : 3s200tq144-4

 Number of Slices:                    1880  out of   1920    97%
 Number of Slice Flip Flops:          2118  out of   3840    55%
 Number of 4 input LUTs:              2971  out of   3840    77%
    Number used as logic:             2418
    Number used as Shift registers:    297
    Number used as RAMs:               256
 Number of IOs:                         62
 Number of bonded IOBs:                 62  out of     97    63%
    IOB Flip Flops:                     64
 Number of BRAMs:                        4  out of     12    33%
 Number of MULT18X18s:                   3  out of     12    25%
 Number of GCLKs:                        4  out of      8    50%
 Number of DCMs:                         1  out of      4    25%

---------------------------
```

**Fig: Synthesis Report**

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 15.304ns (Maximum Frequency: 65.342MHz)
   Minimum input arrival time before clock: 6.569ns
   Maximum output required time after clock: 16.181ns
   Maximum combinational path delay: No path found
```

**Fig: Timing Report**

**Conclusion:**

An efficient non-linear algorithm to remove high-density salt and pepper noise is proposed. The modified sheer sorting architecture educes the computational time required for finding the median. This increases the efficiency of the system. The algorithm removes noise even at higher noise densities and preserves the edges and fine details. The performance of the algorithm is better when compared to the other architecture of this type.

**REFERENCES:**

[I]Xilinx,
http://www.xilinx.comlproducts/design.

[2] Xilinx Inc., Pico Blaze 8-bit Embedded MicrocontrollerUserGuide.http://www.xilinx.comlsupport/documentation/ userJluides/ugl29.

[3]Diligent Inc., Diligent Nexys2 Board Reference Manual

[4] Xilinx. Inc., Platform Specification Format Reference Manual, Embedded Development Kit EDK 9.2i

[6] Xilinx Inc. Xilinx ISE and Xilinx EDK tools.

[7] Spartan-3 Starter Kit Board User Guide, Xilinx, Inc.

[8] Embedded System Tools Reference Manual, Xilinx, Inc

[9] Spartan-3 FPGA Family: Complete Data Sheet

[10] Platform Studio User Guide, Xilinx, Inc.

[11] Xilinx. Inc., Platform Specification Format Reference Manual, Embedded Development Kit EDK 9.2i

[12] Xilinx, Embedded System Example, XAPP433, version 2.2, 2006.