VLSI IMPLEMENTATION OF RANK ORDER FILTERING

By

M.JAISHREE

Reg. No.0920106006

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

A PROJECT REPORT

Submitted to the

FACULTY OF ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

APPLIED ELECTRONICS

APRIL 2011

BONAFIDE CERTIFICATE

Certified that this project report entitled "**VLSI IMPLEMENTATION OF RANK ORDER FILTERING**" is the bonafide work of **Ms. M. Jaishree** [**Reg.no.0920106006**] who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

K. Thilagavathi 6/4/11

Project Guide

Ms.K.Thilagavathi,M.E

Head of the department

Dr.Rajeswari Mariappan,Ph.D

The candidate with University Register No.**0920106006** is examined by us in the project viva-voce examination held on 21.4.2011.....

21/4/11

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

A project of this nature needs co-operation and support from many for successful completion. In this regards, I am fortunate to express my heartfelt thanks to **Padmabhusan Arutselvar Dr.N.Mahalingam B.Sc.,F.I.E.,** Chairman and Co-Chairman **Dr.B.K.Krishnaraj Vanavarayar B.Com,B.L.,** for providing necessary facilities throughout the course.

I would like to express my thanks and appreciation to the many people who have contributed to the successful completion of this project. First I thank **Dr.J.Shanmugam Ph.D,** Director, for providing me an opportunity to carry out this project work.

I would like to thank **Dr.S.Ramachandran Ph.D,** Principal, who gave his continual support and opportunity for completing the project work successfully.

I would like to thank **Dr.Rajeswari Mariappan Ph.D,** Prof of Head, Department of Electronics and Communication Engineering, who gave her continual support for me throughout the course of study.

I would like to thank **Ms.K.Thilagavathi M.E** Assistant Professor(SRG), Project guide for her technical guidance, constructive criticism and many valuable suggestions provided throughout the project work.

My heartfelt thanks to **Ms.R.Latha M.E(Ph.D),** Associate. Professor , Project coordinator, for her contribution and innovative ideas at various stages of the project to successfully complete this work.

I express my sincere gratitude to my family members, friends and to all my staff members of Electronics and Communication Engineering Department for their support throughout the course of my project.

ABSTRACT

Two applications of great importance in the area of image processing are noise filtering and image enhancement. These tasks are an essential part of any image processor whether the final image is utilized for visual interpretation or for automatic analysis. The aim of noise filtering is to eliminate noise and its effects on the original image, while corrupting the image as little as possible. To this end, nonlinear techniques (like the median and, in general, order statistics filters) have been found to provide more satisfactory results in comparison to linear methods. Rank-order filtering (ROF), or order statistical filtering, has been widely applied for various speech and image processing applications. This paper addresses on VLSI design of Rank-Order Filtering (ROF) with a maskable memory, it neither sorts/swaps nor modifies the window samples and requires less hardware resources. The ROF algorithm has low computational complexity and can map to highly parallel architecture called DCRAM. The maskable registers allow the architecture to selectively read or write bit-slices, and hence speed up "parallel read" and "parallel polarization". The proposed architecture offer good area, noise ratio and speed improvement. The architecture is designed using VHDL and functional verification is done using Modelsim.

LIST OF FIGURES

LIST OF ABBREVIATIONS

ROF	- Rank order Filter
DCRAM	- Dual Cell Random Access Memory
IIR	- Infinite Impulse Response
AMF	- Adaptive Median Filter
CAM	- Content Addressable Access Memory
SRAM	- Static Random Access Memory
FPGA	- Field Programmable Gate Array
RMF	- Recursive Median Filter

CHAPTER -1
INTRODUCTION

Rank-Order Filtering (ROF), or order-statistical filtering, has been widely applied for various speech and image processing applications. Unlike linear filtering, ROF can remove sharp discontinuities of small duration without blurring the original signal; therefore, ROF becomes a key component for signal smoothing and impulsive noise elimination. To provide this key component for various signal processing applications, we intend to design a configurable Rank-Order Filter that features low cost and high speed.

1.1 MOTIVATION:

Many approaches for hardware implementation of ROF have been presented previously. Many of them are based on sorting algorithm. They considered the operation of ROF as two steps: sorting and choosing. The systolic architectures for ROF based on sorting algorithms, such as bubble sort and bitonic sort. These architectures are fully pipelined for high throughput rate at the expense of latency, but require a large number of compare-swap units and registers.

1.1.1. DELETE AND INSERT SORT ALGORITHM:

Delete-and insert sort algorithm is one of the many available sort algorithms. Its operations can be described as follows. For data insertion as shown in Fig. 1, the current input sample is known to be 20 and should be placed in between 25 and 16 which are already sorted and stored in a memory. Each time a new sample is given, it can be placed according to this scheme. After all samples are processed, the sorted sequence can be obtained from the memory. For data deletion, it only needs to identify the sorted item whose value is equal to the input sample's value as shown in Fig. 1. Thus for any running order operation, this delete operation becomes useful since only those samples out of the mask region need to be removed and new input samples need to be sorted. In addition to

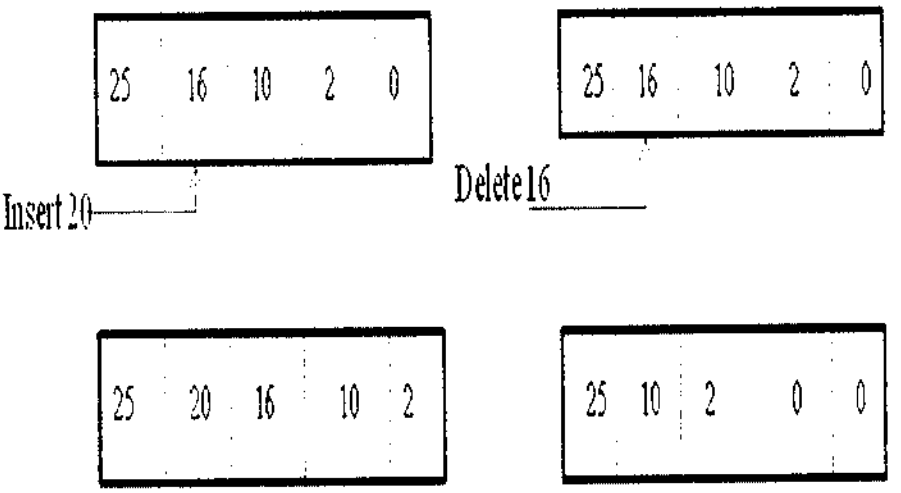

Fig 1.1 Delete and insert sort algorithm

these features, this algorithm can also provide an ascending sequence if smaller items are placed at left side instead of right side as used for a descending sequence (as illustrated in Fig.1.1). This algorithm description implies that a lot of comparisons are needed in order to allocate a position in which the input sample can be correctly placed or removed. Moreover a lot of move operations are needed before the input sample is inserted or deleted. For real-time image video processing, either insert or delete operation has to be done in a very stringent timing constraint. Thus if this algorithm is implemented on a general purpose computer, the result is obviously not suitable for real-time applications and hence the algorithm has to be modified.

Therefore, memory architecture for Rank Order Filter based on a generic ROF algorithm is designed. The generic ROF algorithm uses threshold decomposition to bitwisely determine the rank-order result from the most significant bit (MSB) to the least significant bit (LSB) and applies the polarization simultaneously to polarize impossible candidates. In order to improve the efficiency Adaptive Median Filter is designed.

1.2 OVERVIEW OF ROF:

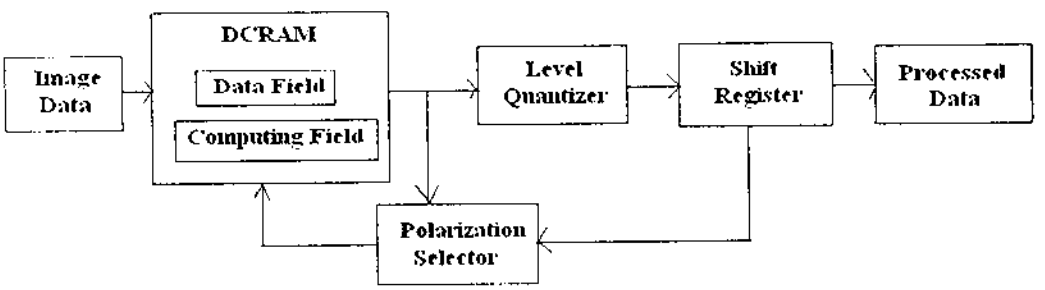

Fig 1.2 Block diagram of ROF

1.2.1 DESIGN STEPS OF RANK ORDER FILTER:

Step 1: The input data is given with window size **N**, bit width **B**, rank **r**.

Step 2: The data fetched first loads the input samples to **data field** and then it makes copy from data field to **computing field**.

Step 3: The maskable part of DCRAM performs parallel read for 'bit counting' and parallel writes for 'polarization'.

Step 4: The bit-sliced values then goes to **level quantizer** for 'threshold decomposition'.

Step 5: The ROF performs 'polarization' using **polarization selector** to polar the lower bits.

Step 6: After executing **B** times of the three main task and reach the LSB of the ROF. The result will be in **shift register.**

Step 7: The **processed data** is obtained after a cycle is completed.

1.3 ORGANISATION OF REPORT

Chapter 2: Explains noise, its generation and its types.

Chapter 3: Deals with filters and its types.

Chapter 4: Explains about Rank Order Filter and its algorithm.

Chapters 5: Deals with ROF architecture.

Chapter 6: Explains about Adaptive Median Filter design

Chapter 7: Discuss about simulation results

CHAPTER -2
NOISE

2.1 NOISE IN IMAGES:

Image noise is the random variation of brightness or color information in images produced by the sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector.

Image noise is generally regarded as an undesirable by-product of image capture. Although these unwanted fluctuations became known as "noise" by analogy with unwanted sound, they are inaudible and actually beneficial in some applications, such as dithering.

2.2 NOISE GENERATION:

Real signals usually contain departures from the ideal signal that would be produced by our model of the signal production process. Such departures are referred to as noise. Noise arises as a result of unmodelled or unmodellable processes going on in the production and capture of the real signal. It is not part of the ideal signal and may be caused by a wide range of sources, e.g. variations in the detector sensitivity, environmental variations, the discrete nature of radiation, transmission or quantization errors, etc.

The characteristics of noise depend on its source, as does the operator which best reduces its effects. Many image processing packages contain operators to artificially add noise to an image. Deliberately corrupting an image with noise allows us to test the resistance of an image processing operator to noise and assess the performance of various noise filters.

2.3 NOISE IN DIGITAL IMAGES:

"Image noise" is the digital equivalent of film grain for analog cameras. Alternatively, one can think of it as analogous to the subtle background hiss one may hear from an audio system at full volume. For digital images, this noise appears as random speckles on an otherwise smooth surface and can significantly degrade image quality.

Images taken with both digital cameras and conventional film cameras will pick up noise from a variety of sources. Many further uses of these images require that the noise will be (partially) removed - for aesthetic purposes as in artistic work or marketing, or for practical purposes such as computer vision.

2.4 TYPES OF NOISE IN DIGITAL IMAGES:

- Shot noise
- Salt and pepper noise
- Gaussian noise.
- Quantization Noise

2.4.1 Shot Noise

The dominant noise in the lighter parts of an image from an image sensor is typically that caused by statistical quantum fluctuations, that is, variation in the number of photons sensed at a given exposure level; this noise is known as photon shot noise. Shot noise has a root-mean-square value proportional to the square root of the image intensity, and the noises at different pixels are independent of one another. Shot noise follows a Poisson distribution, which is usually not very different from Gaussian.

In addition to photon shot noise, there can be additional shot noise from the dark leakage current in the image sensor; this noise is sometimes known as "dark shot noise" or "dark-current shot noise". Dark current is greatest at "hot pixels" within the image sensor; the variable dark charge of normal and hot pixels can be subtracted off (using "dark frame subtraction"), leaving only the shot noise, or random component, of the leakage; if dark-frame subtraction is not done, or if the exposure time is long enough that

the hot pixel charge exceeds the linear charge capacity, the noise will be more than just shot noise, and hot pixels appear as salt-and-pepper noise.

2.4.2. Salt and Pepper Noise

Salt and pepper noise is a form of noise typically seen on images. It represents itself as randomly occurring white and black pixels. An effective noise reduction method for this type of noise involves the usage of a median filter. Salt and pepper noise creeps into images in situations where quick transients, such as faulty switching.

2.4.3 Gaussian Noise:

In Gaussian noise each pixel in the image will be changed from its original value by a (usually) small amount. A histogram, a plot of the amount of distortion of a pixel value against the frequency with which it occurs, shows a normal distribution of noise. While other distributions are possible, the Gaussian (normal) distribution is usually a good model, due to the central limit theorem that says that the sum of different noises tends to approach a Gaussian distribution.

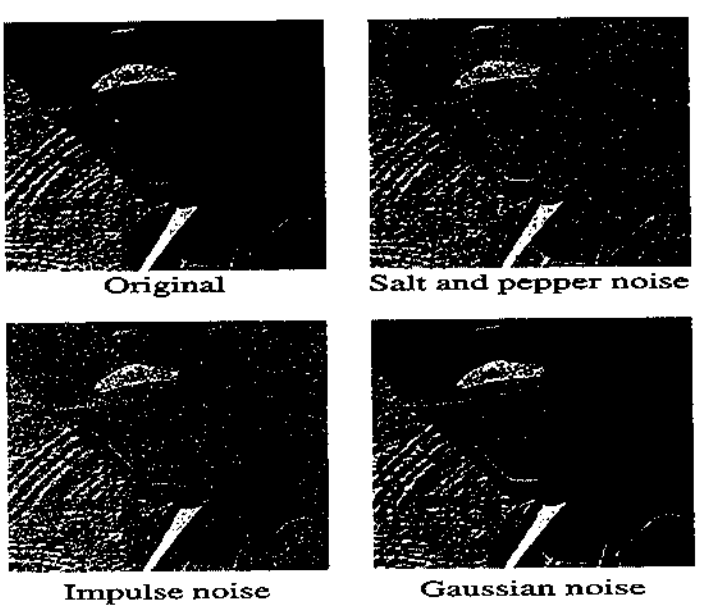

Fig.2.1Example for types of noise

2.4.4 Quantization Noise:

The noise caused by quantizing the pixels of a sensed image to a number of discrete levels is known as quantization noise; it has an approximately uniform distribution, and can be signal dependent, though it will be signal independent if other noise sources are big enough to cause dithering, or if dithering is explicitly applied.

2.5 IMAGE NOISE REDUCTION:

Most algorithms for converting image sensor data to an image, whether in-camera or on a computer, involve some form of noise reduction. There are many procedures for this, but all attempt to determine whether differences in pixel values constitute noise or real photographic detail, and average out the former. However, no algorithm can make this judgment perfectly, so there is often a tradeoff made between noise removal and preservation of fine, low-contrast detail that may have characteristics similar to noise. Many cameras have a setting to control the aggressiveness of the in-camera noise reduction.

This decision can be assisted by knowing the characteristics of the source image and of human vision. Most noise reduction algorithms perform much more aggressive chroma noise reduction, since there is little important fine chroma detail that one risks losing. Furthermore, many people find luminance noise less objectionable to the eye, since its textured appearance mimics the appearance of film grain.

CHAPTER 3
FILTERS

A filter is a linear time- invariant system used for removing undesirable noise from desired signals.It can be used in spectral shaping such as equalization of communication channels, signal detection in radar, sonar and communication.

3.1 LINEAR FILTER

Linear filters in the time domain process time-varying input signals to produce output signals, subject to the constraint of linearity. This results from systems composed solely of components (or digital algorithms) classified as having a linear response.

Most filters implemented in analog electronics, in digital signal processing, or in mechanical systems are classified as causal, time invariant, and linear. However the general concept of linear filtering is broader, also used in statistics, data analysis, and mechanical engineering among other fields and technologies. This includes non causal filters and filters in more than one dimension such as would be used in image processing; those filters are subject to different constraints leading to different design methods, which are discussed elsewhere.

A linear time-invariant (LTI) filter can be uniquely specified by its impulse response h, and the output of any filter is mathematically expressed as the convolution of the input with that impulse response. The frequency response, given by the filter's transfer function $H(\omega)$, is an alternative characterization of the filter. The frequency response may be tailored to, for instance, eliminate unwanted frequency components from an input signal, or to limit an amplifier to signals within a particular band of frequencies..

Among the time-domain filters we here consider, there are two general classes of filter transfer functions that can approximate a desired frequency response. Very different mathematical treatments apply to the design of filters termed infinite impulse response

(IIR) filters, characteristic of mechanical and analog electronics systems, and finite impulse response (FIR) filters, which can be implemented by discrete time systems such as computers .

Examples of linear filter

- triangular filter
- Gaussian filter
- exponential filter
- Kalman filter
- Smoother (Smoothing Filter)
- predicting filter

3.2 NON-LINEAR FILTER

A **nonlinear filter** is a signal-processing device whose output is not alinear function of its input. Terminology concerning the filtering problem may refer to the time domain representation of the signal or to the frequency domain representation of the signal. When referring to filters with adjectives such as "bandpass, highpass, and lowpass" one has in mind the frequency domain. When resorting to terms like "additive noise", one has in mind the time domain, since the noise that is to be added to the signal is added in the state space representation of the signal. The state space representation is more general and is used for the advanced formulation of the filtering problem as a mathematical problem in probability and statistics of stochastic processes.

Nonlinear filters locate and remove data that is recognised as noise. The algorithm is 'nonlinear' because it looks at each data point and decides if that data is noise or valid signal. If the point is noise, it is simply removed and replaced by an estimate based on surrounding data points, and parts of the data that are not considered noise are not modified at all. Linear filters, such as those used in bandpass, highpass, and lowpass, lack such a decision capability and therefore modify all data. Nonlinear filters are sometimes used also for removing very short wavelength, but high amplitude features from data.

Such a filter can be thought of as a noise spike-rejection filter, but it can also be effective for removing short wavelength geological features, such as signals from surficial features.

Examples of nonlinear filters include:

- phase-locked loops
- detectors
- mixers
- median filters
- ranklets

CHAPTER-4
RANK ORDER FILTER

4.1 INTRODUCTION:

Rank order filters are non-linear filters used in a wide range of applications. The median filter is perhaps the most common of the rank order filters. It can be used to remove noise from a signal while preserving edge information which is something that linear filters cannot do. This makes the median filter a popular filter in speech and image processing applications. Given a sequence of input samples, the basic operation of ROF is to choose the rth largest sample as the output Y, where R is the rank-order of the filter .

Rank order filters work by sorting a window of inputs and selecting the input with a certain rank as the output. In the case of the median filter the element with the middle rank is selected, hence the name (although it is called a speckle filter in some applications, after the type of noise it removes). Other common rank order filters are the minimum and maximum filter, selecting respectively the input with the lowest and highest rank from the window. In image processing applications these minimum and maximum filters are sometimes also called dilation and erosion filters, because they make dark regions in an image respectively larger or smaller.

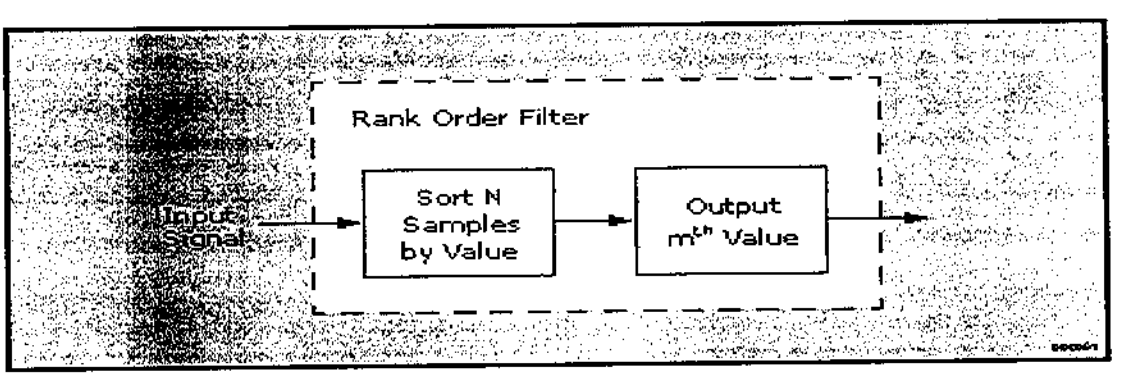

Fig 4.1 Rank Order Filter

4.2 TYPES OF ROF:

There are two types of ROF present. They are

- Recursive ROF
- Non- Recursive ROF

4.2.1 RECURSIVE ROF:

In signal processing, a **recursive filter** is a type of filter which re-uses one or more of its outputs as an input. This feedback typically results in an unending impulse response (commonly referred to as infinite impulse response(IIR)), characterised by either exponentially growing, decaying or sinusoidal signal output components. However, a recursive filter does not always have an infinite impulse response. Some implementations of moving average filter are recursive filters but with a Finite impulse response. A recursive filter is one which in addition to input values also uses previous output values. These, like the previous input values, are stored in the processor's memory. The word recursive literally means "running back", and refers to the fact that previously-calculated output values go back into the calculation of the latest output.

4.2.1 NON- RECURSIVE ROF:

The current output (y_n) is calculated solely from the current and previous *input* values (x_n, x_{n-1}, x_{n-2}, ...). This type of filter is said to non-recursive. A major advantage of recursive filters over **non-recursive filter** is that they are computationally simpler. The non-recursive variant may be characterized by an infinitely large window size and, hence, be difficult for practical implementation.

4.3 MEDIAN FILTER:

Median filters are often used to remove impulse noise in image processing. In signal processing, it is often desirable to be able to perform some kind of noise reduction on an image or signal. The median filter is a nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise. Its performance for speckle noise and salt and pepper noise (impulsive noise), it is particularly effective. Because of this, median filtering is very widely used in digital image processing.

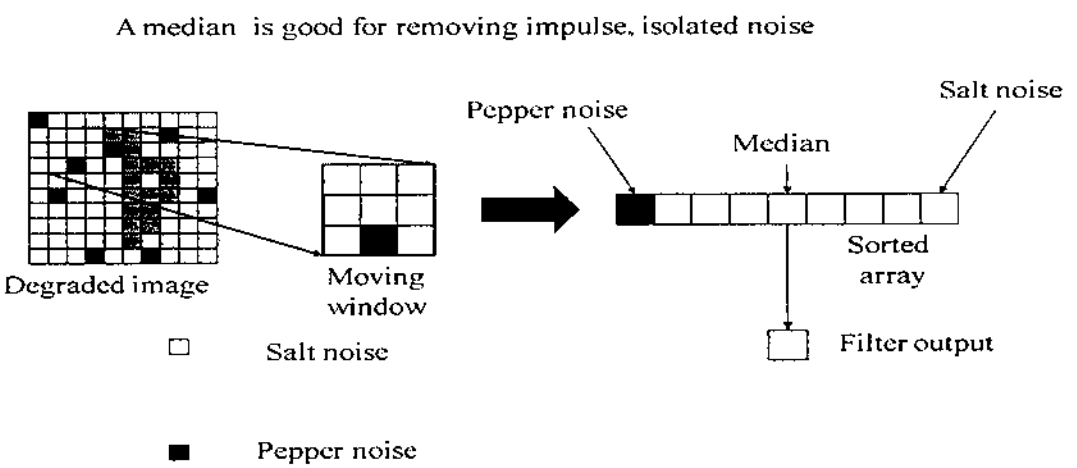

Fig.4.2 Median Filter

4.4 ROF ALGORITHM

4.4.1 BIT SLICED ROF ALGORITHM:

This project presents novel memory architecture for ROF based on a generic ROF algorithm. The generic ROF algorithm uses the threshold decomposition to bit wisely determine the rank-order result from the most significant bit (MSB) to the least significant bit (LSB) and applies the polarization simultaneously to polarize impossible candidates. Using this algorithm, the result are taken bit-by-bit, for a specific rank- order without sorting the numbers. Note that the sorting is the most complex part in conventional ROF implementations. Basically, there are three major tasks in the generic algorithm: they are **parallel read, threshold decomposition**, and **parallel polarization**. This project presents a maskable memory structure, motivated from CAM architecture, to realize these tasks efficiently. The maskable memory structure, called Dual-Cell Random-Access Memory (DCRAM), is an extended SRAM structure with maskable registers and dual cells. The maskable registers allow the architecture to selectively read or write bit-slices, and hence speed up ``parallel and ``parallel polarization'' tasks.

4.4.2 STEPS FOR ROF ALGORITHM:

Given the input samples, the window size N= l+k+1, the bit-width B and the rank r:

Step 1: Set b= B-1.

Step 2 (Bit Counting): Calculate Z_b from $\{ u^b_{i-k}, u^b_{i-k+1}, \ldots u^b_i \ldots \ldots u^b_{i+1} \}$.

Step 3(Thershold decomposition): If $z_b \geq r$, $v^b_i = 1$; otherwise $v^b_i = 0$.

Step 4 (Polarization): If $u^b_j \neq v^b_i$, $u^m_j = u^b_j$ for $0 \leq m \leq b-1$ and $i-k \leq j \leq i+l$.

Step 5 : b=B-1.

Step 6 : If $b \geq 0$ go to Step2.

Step 7: Output y_i

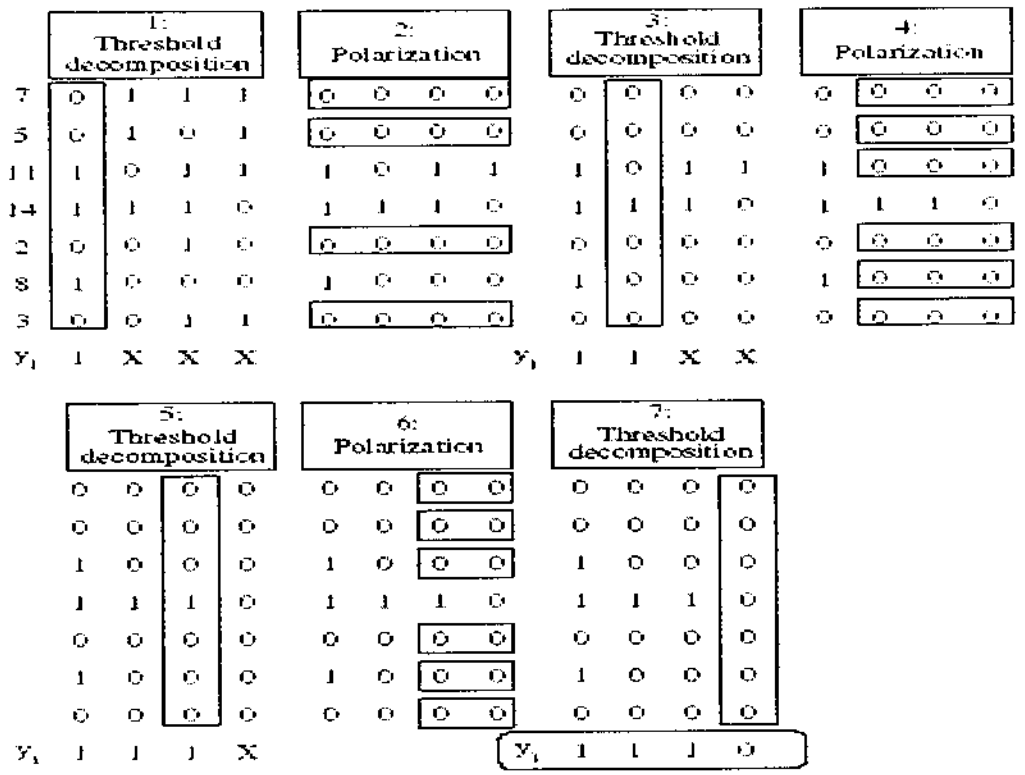

Fig 4.3 Example of generic bit sliced ROF algorithm for N=7 B=4 r=1

The steps shown below will explain as how the result is obtained:

STEP 1: "Bit counting" step will calculate the number of 1's at MSBs, which is 3.

STEP 2: The number of 1's is greater than r, the "Threshold decomposition" step sets the MSB of y_i to '1'.

STEP 3: Then, the "Polarization" step will consider the inputs with $u^3_j=1$ as candidates of the ROF output and polarize the lower bits of the others to all 0's.

STEP 4: After repeating the above steps with decreasing b, the output y_i will be 14(1110

CHAPTER -5
ROF ARCHITECTURE

5.1 The dual-cell RAM architecture for ROF:

The dual-cell random-access memory (DCRAM) plays a key role in the proposed ROF architecture. In the DCRAM, there are two fields for reusing the input data and pipelining the filtering process. For the one-dimensional (1-D) ROF, the proposed architecture receives one sample at a time.

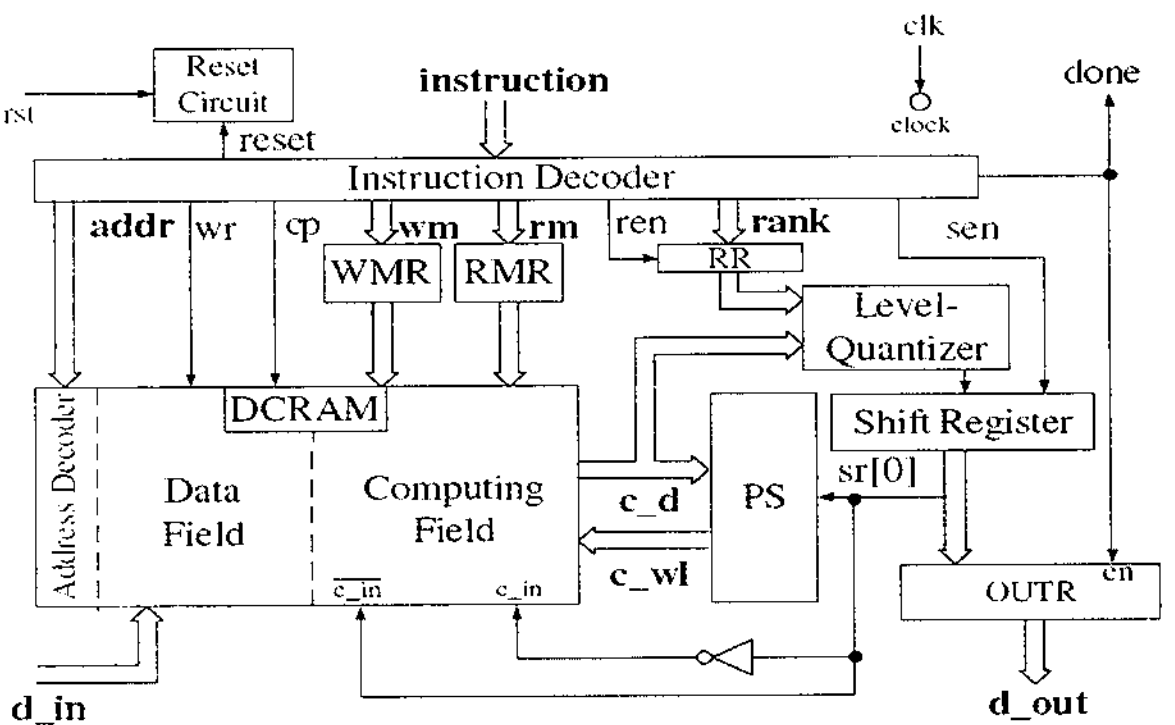

Fig 5.1 The proposed Rank Order Filtering architecture

For the n-by-n two-dimensional (2-D) ROF, the architecture reads n samples into the input window within a filtering iteration. To speed up the process of ROF and pipeline the data loading and filtering calculation, the data field loads the input data while the computing field is performing bit-sliced operations. Hence, the execution of the architecture has two pipeline stages: data fetching and rank-order calculation

The computing field is in the maskable part of DCRAM. The maskable part of DCRAM performs parallel reads for bit counting and parallel writes for polarization. The read mask register (RMR) is configured to mask unwanted bits of the computing field during read operation. The value of RMR is one-hot-encoded so that the bit-sliced values can be read from the memory in parallel. The bit-sliced values will then go to the level-quantizer for threshold decomposition.

When the ROF performs polarization, the write mask register (WMR) is configured to mask untouched bits and allow the polarization selector (PS) to polar lower bits of non-candidate samples. Since the structure of memory circuits is regular and the maskable scheme provides fast logic operations, the maskable memory structure features low cost and high speed.

5.2 LEVEL-QUANTIZER:

The level-quantizer performs 'bit-counting' and 'thershold decomposition' by summing up bits of the bit sliced value to Z_b and comparing Z_b with the rank value r. The rank value r is stored in the rank register (RR). The block diagram of the level-quantizer, where FA denotes the full adder and HA denotes the half adder. The signals"S" and "C" of each FA or HA represent sum and carry, respectively. The circuit in the dash-lined box is a comparator. The comparator is implemented by a carry generator because the comparison result of Z_b and r can be obtained from the carry output of Z_b plus the two's complement of r. The carry output is the quantized value of the level-quantizer.

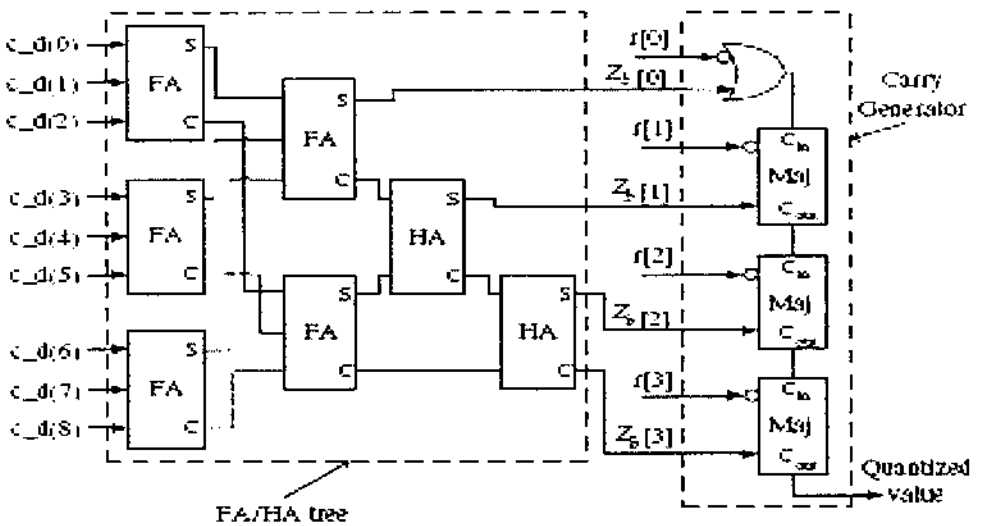

Fig 5.2 The block diagram of level-quantizer

After the level-quantizer finishes the threshold decomposition the quantized values go to the LSB of the shift register, "SR[0]". The polarization selector (PS) uses exlusive Ors to determine which word should be polarized.

5.3 POLARIZATION SELECTOR:

After the level-quantizer finishes the threshold decomposition, the quantized value goes to the LSB of the shift register, ''sr[0]''. Then, the polarization selector (PS) uses exclusive ORs (XORs) to determine which words should be polarized, as shown in Fig. 4. Obviously, the XORs can examine the condition of $u^b_j \neq v^b_i$ and select the word-under polarization's (WUPs) accordingly. When ''c_wl'' is ''1'', the lower bits of selected words will be polarized; the lower bits are selected by WMR. According to the Step 4, the polarized value is u^b_j which is the inversion of v^b_i . Since v^b_i is the value of sr[0], we inverse the value of ''sr[0]'' to ''c_in'', as shown in Fig. 2.

As seen in the generic algorithm, the basic ROF repeatedly executes bit-counting, threshold decomposition, and polarization until the LSB of the ROF result being

generated. Upon executing B times of three main tasks, the ROF will have the result in the shift register. A cycle after, the result will then go to the output register (OUTR).

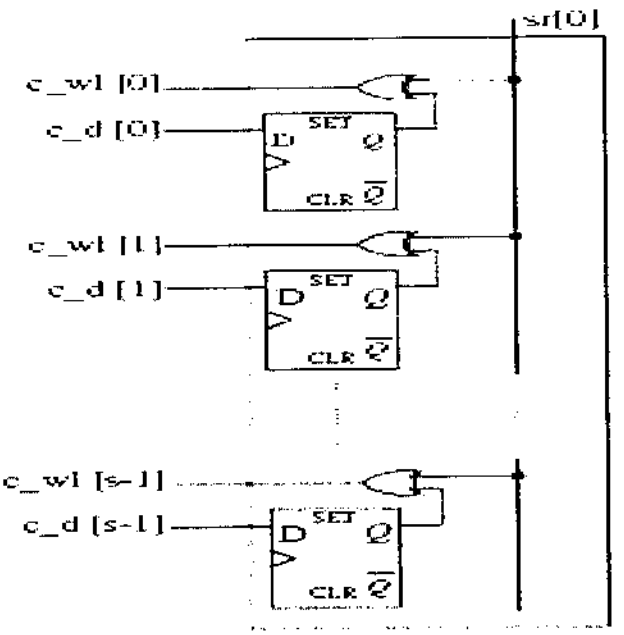

Fig.5.3 The block diagram of polarization selector

Doing so, the proposed architecture is able to pipeline the iterations for high-performance applications.

On the bit-sliced algorithm, the bit-serial logic network can bitwisely select the ranked candidates and generates the ranked result one bit at a time. The bit-serial logic network recursively executes two steps: majority calculation and polarization. This type of ROFs can reduce the latency complexity to O(B); however, many of them use complex logic networks to implement the majority calculation. In earlier work they uses an inverter as a voter for majority calculation. It significantly improves both hardware cost and processing speed; nevertheless, the noise margin will become narrow as the number of inputs increases. The proposed architecture basically takes the advantages of the bit-sliced algorithm: (1) the latency is independent of the window size, and (2) the result can be obtained without exhaustive comparisons. Comparing with the bit-serial logic

network, the proposed architecture has higher degree of flexibility and regularity because of the DCRAM structure. The DCRAM structure reduces not only the complexity of the majority calculation, but also the routing area between components. Another major strength of the proposed ROF processor is the programmability. The programmable ROF processor; however, their application is limited to median filtering. The DCRAM structure, the proposed ROF processor is flexible with the variation of the rank order r and algorithms. ROF processor can be programmed for any rank order r and diverse applications. Furthermore, the proposed architecture can reuse input data as many as possible and hence reduce the power consumption on memory access.

5.4 APPLICATION OF ROF PROCESSOR:

In this section we use 1-D non-recursive ROF as an example to show the programming of the proposed ROF processor. Due to the programmable design, the proposed ROF processor can implement a variety of ROF applications. The following subsections will illustrate the optimized programs for three examples: 1-D RMF, 2-D non recursive ROF, and 2-D RMF.

5.4.1 1-D recursive median filter

The recursive median filtering (RMF) has been proposed for signal smoothing and impulsive noise elimination. It can effectively remove sharp discontinuities of small duration without blurring the original signal. The RMF recursively searches for the median results from the most recent median values and input samples.

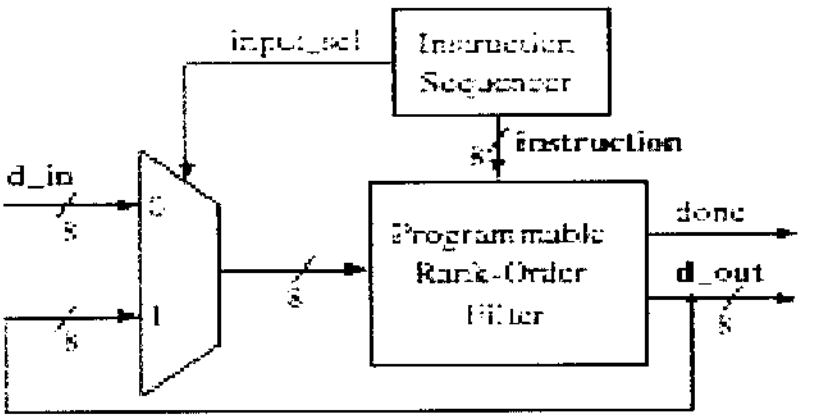

Fig 5.4 The block diagram of 1-D RMF

Fig 5.4 perform RMF with previous median values, the ith iteration of 1-D RMF loads two inputs to the DCRAM; one is $x_i^{p}1$ and the other is y_i_1. As shown in Fig. 5.4, the 2-to-1 multiplexer is used to switch the input stream to the data field, controlled by the instruction sequencer; the input stream is from either "d_in" or "d_out".

5.4.2 2-D non-recursive rank-order filter

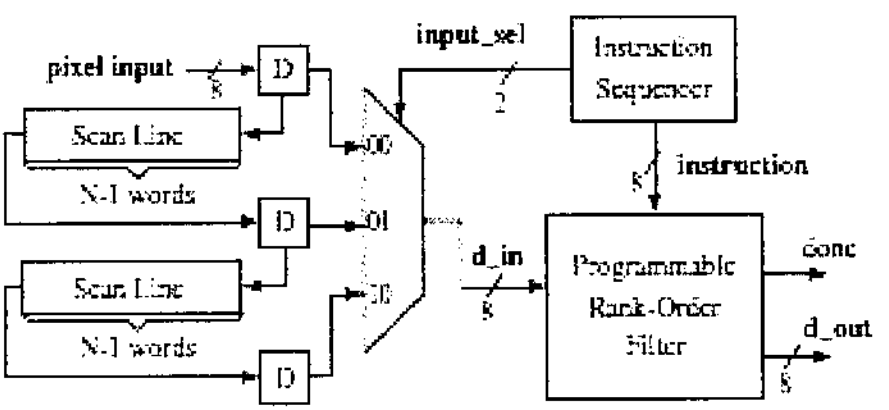

Fig 5.5 Block diagram of 2-D non recursive ROF with 3 by 3 window

Fig. 5.5 illustrates the block diagram for the 2-D non recursive ROF. From Fig. 5.6, each iteration needs to update three input samples (the pixels in the shadow region)

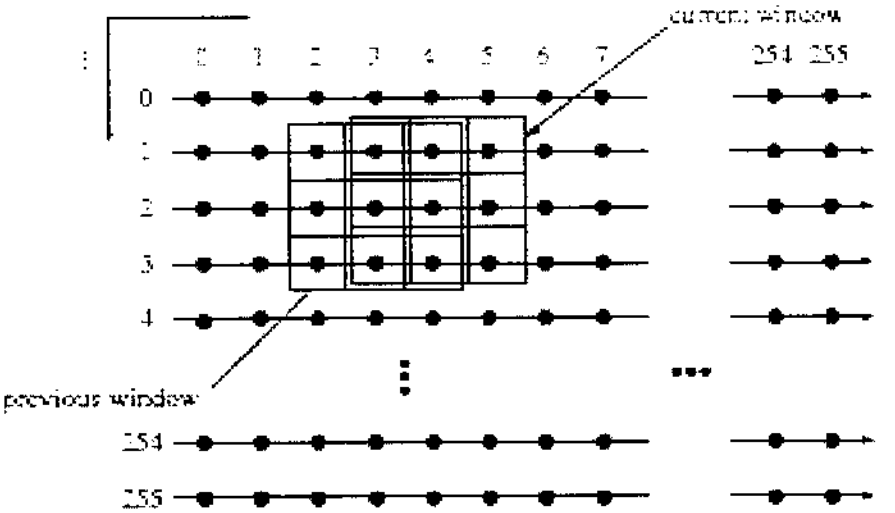

Fig 5.6 The windowing of 3 x 3 non recursive ROF

for the 3 x 3 ROF; that is, only n input samples need to be updated in each iteration for the n x n ROF. So, for the 2-D ROF, the data reusability of our process is high; each iteration updates only n input samples for an n x n window.

5.4.3 2- D recursive median filter

Similar to the 1-D RMF, the 2-D n -by-n RMF finds the median value from the window formed by some previous previous calculated median values and input values. At the end of each iteration, the 2-D 3 x 3 RMF substitutes the central point of the current window with the median value. The renewed point will then be used in the next iteration. The windowing for 2-D RMF iterations is previous calculated median values and the pixels in the shadow region are updated at the beginning of each iteration.

5.5 ADVANTAGES:

- Noise Suppression
- Edge detection
- Edge enhancement
- Smooth noise

5.6 APPLICATIONS :

- Speech signal processing
- Image processing
- Medical image processing

The basic disadvantage of the application of the median filter is the blurring of the image in process. In the general case, the filter is applied uniformly across an image, modifying pixels that are not contaminated by noise. In this way, the pixel values of the input image are altered, leading thus to an overall degradation of the image and to blurred or distorted features.

Therefore an adaptive approach is implemented to improve image quality at the output. The main advantage of this adaptive approach is that the blurring of the image in process is avoided and the integrity of edge and detail information is preserved. Moreover, the utilization of the median filter is done in a more efficient way.

CHAPTER -6
ADAPTIVE MEDIAN FILTER

6.1 INTRODUCTION:

The Adaptive Median Filter is designed to eliminate the problems faced with the standard median filter. The basic difference between the two filters is that, in the Adaptive Median Filter, the size of the window surrounding each pixel is variable. This variation depends on the median of the pixels in the present window. If the median value is an impulse, then the size of the window is expanded. Otherwise, further processing is done on the part of the image within the current window specifications. 'Processing' the image basically entails the following: The center pixel of the window is evaluated to verify whether it is an impulse or not. If it is an impulse, then the new value of that pixel in the filtered image will be the median value of the pixels in that window. If, however, the center pixel is not an impulse, then the value of the center pixel is retained in the filtered image. Thus unless the pixel being considered is an impulse, the gray-scale value of the pixel in the filtered image is the same as that of the input image. Thus, the Adaptive Median Filter solves the dual purpose of removing the impulse noise from the image and reducing distortion in the image. Adaptive Median Filtering can handle the filtering operation of an image corrupted with impulse noise of probability greater than 0.2. This filter also smoothens out other types of noise, thus, giving a much better output image than the standard median filter.

6.2 DESIGN OF ADAPTIVE MEDIAN FILTER:

The most common method used for impulse noise suppression for gray-scale and color images is the median filter. Impulse noise exists in many practical applications and can be generated by various sources, including many man-made phenomena such as unprotected switches, industrial machines, and car ignition systems. Images are often corrupted by impulse noise due to a noisy sensor or channel transmission errors. This type of noise is the classical salt-and-pepper noise for grayscale images.

The output of a median filter at a point x of an image f depends on the values of the image points in the neighborhood of x. This neighborhood is determined by a window W that is located at point x of f including n points $x_1, x_2, \ldots, x_n$ of f. the median can be determined when the number of points included in W is odd i.e., when $n = 2k+1$. the n values $f(x_1), f(x_2), \ldots, f(x_n)$ of the n points $x_1, x_2, \ldots, x_n$ are placed in ascending order forming the set of ordered values $\{f_1, f_2, \ldots, f_n\}$, in which $f_1 \leq f_2 \leq \ldots, \leq f_n$. The median is defined as he (k+1)th value of the set $\{f_1, f_2, \ldots, f_n\}$, med = f_{k+1}.

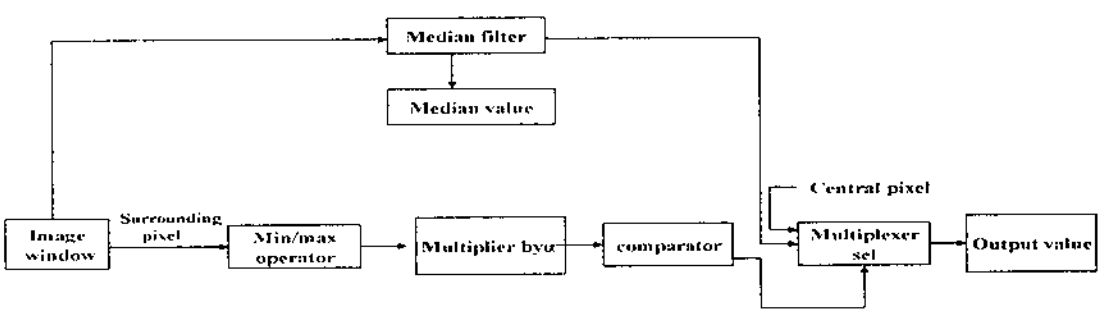

Fig. 6.1 Adaptive Median Filter

The main advantage of this adaptive approach is that the blurring of the image in process is avoided and the integrity of edge and detail information is preserved. Moreover, the utilization of the median filter is done in a more efficient way. Experimental results demonstrate the improved performance of the AMF. The proposed digital hardware structure is capable of processing gray-scale images of 8-bit resolution and performs both positive and negative impulse noise removal. A moving window of a 3x3 and 5x5 pixel image neighborhood can be selected. Furthermore, the system is directly expandable to accommodate larger size gray-scale

images. The architecture chosen is based on a sequence of four basic functional pipelined stages and parallel processing is used within each stage. The proposed structure was implemented using field-programmable gate arrays (FPGAs) , which offer an attractive combination of low-cost, high-performance, and apparent flexibility. The total number of the system inputs and outputs are 44 and eight pins, respectively and the percentage of the logic cells utilized is (40 inputs for the input data and four in- puts for the clock and the control signals required) and the percentage of the logic cells utilized is 99%. The typical clock frequency is 65 MHz and the system can be used for real-time imaging applications where fast processing is of the utmost importance. As an example, the time required to perform filtering of a grayscale image of 260x244 pixels is approximately 7.6 ms.

The function of the proposed circuit is to detect first the existence of noise in the image window and apply the corresponding median filter only when necessary. The noise detection level procedure can be controlled so that a range of pixel values (and not only the fixed values 0 and 255, but also salt-and-pepper noise) is considered as impulse noise. The main advantage of this adaptive approach is that the blurring of the image in process is avoided and the integrity of edge and detail information is preserved. Moreover, the utilization of the median filter is done in a more efficient way.

6.3 STEPS FOR ADAPTIVE MEDIAN FILTER:

The proposed adaptive median filter can be utilized for impulse noise suppression for gray-scale images. Its function is to detect the existence of noise in the image window and apply the corresponding median filter only when necessary. The steps as follows:

1) For a neighborhood window W that is located at point x of the image f, the maximum (minimum) pixel value of the n-1 surrounding points of the neighborhood is computed, denoted as $f_{max}(x)$ ($f_{min}(x)$), excluding the value of the central pixel at point x.

2) The value $f_{max}(x)$ ($f_{min}(x)$) is multiplied by a parameter α which is a real number and an be modified. The result is the threshold value for the detection of a noise pixel, denoted as $f_{threshold}(x)$ and is limited to a positive (negative) integer threshold value.

3) The value of the central pixel is compared to $f_{threshold}(x)$, and the central pixel is considered to be noise, when its value is greater (less) than the threshold value $f_{threshold}(x)$.

4) When the central pixel is considered to be noise, it is substituted by the median value of neighbourhood, f_{k+1} which is normal operation of median filter. In the opposite case, the opposite value of central pixel is not altered and the procedure is repeated all the for neighbourhood window.

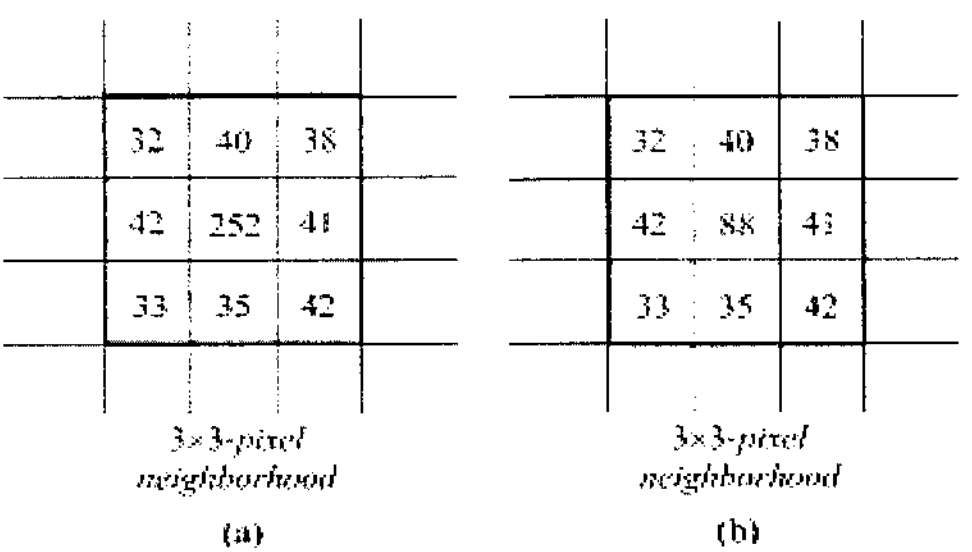

Fig 6.2 Noise detection algorithm (a) Impulse noise (b) Signal-dependent noise

A block diagram of the adaptive filtering procedure previously described is depicted in Fig.6.1 An example of the application of the noise detection algorithm for the cases of impulse and signal dependent noise is illustrated in Fig. 6 (a) and (b), respectively. In Fig.6.2 (a), a typical 3x3 pixel neighborhood window of a gray-scale image is depicted. The central pixel of the window occupies an extreme value (pixel value = 252) compared to the values of the surrounding image points (pixel values ranging from 32 to 42). For this reason, the central pixel is considered to be impulse noise and it should be eliminated. In the specific example, the noise detection scheme is applied as follows. In the first step, we find the maximum value of the surrounding pixels, $f_{max}(x)$ = 42. If we consider the parameter $\alpha = 5$, then threshold value $f_{threshold}(x) = f_{max}(x) * 5 =$ 210. The central pixel value is $252 > f_{threshold}(x)$ and, therefore, is considered to be a noisy pixel. Finally, it is substituted by the median value of the neighborhood. The same discussion applies to the example of Fig.6.2 (b), which is the case of signal-dependent noise. The same procedure is followed, and the noisy pixel is successfully detected for a parameter value $\alpha = 2$.

Two major remarks about the presented adaptive algorithm should be made. First, the value of the parameter α is of great importance, since it controls the operation of the circuit and the result of the overall procedure for different noise cases. Second, an appropriate positive and negative threshold value must be utilized for the case of impulse noise, when $f_{threshold}(x) \geq 255$. For example, in the case of Fig.6.2(a), if we consider the parameter $\alpha = 8$, then $f_{threshold}(x) = f_{max}(x) * 8 = 336$ and the $f_{threshold}(x)$ is limited to the value 255, $f_{threshold}(x) = 255$. The central pixel value is $252 < f_{threshold}(x)$ and the central pixel is erroneously not considered to be impulse noise. An adjustable positive threshold value (for example 240) can be used as a limit of $f_{threshold}(x)$. In this way, $f_{threshold}(x) = 240$, whereas the central pixel value is $252 > f_{threshold}(x)$, and the central pixel is successfully detected as impulse noise. The meaning of this normalization procedure is that pixels occupying values between a range of the impulsive values (and not only pixels with values 0 and 255) should be considered as noisy pixels.

The proposed circuit detects the existence of noise in the image neighborhood and applies the corresponding median filter only when it is necessary. The noise detection procedure is controllable, and, thus, pixel values other than the two extreme ones can be considered as impulse noise, provided that they are significantly different from the central pixel value. In this way, the blurring of the image is avoided.

The system is suitable for real-time imaging applications where fast processing is required. Moreover, the design of the circuit can be easily modified to accommodate larger size windows. The proposed digital hardware structure was designed, compiled and successfully simulated in FPGAs.The typical system clock frequency is 44.835MHz.

CHAPTER -7
RESULTS AND DISCUSSION

The simulation of this project has been done using, MODELSIM XE III6.2g and XILINX XE 9.2i

Modelsim provides a comprehensive simulation and debug environment for complex ASIC and FPGA designs. Support is provided for multiple languages including Verilog, SystemVerilog, VHDL and SystemC.

Xilinx was founded in 1984 by two semiconductor engineers, Ross Freeman and Bernard Vonderschmitt, who were both working for integrated circuit and solid-state device manufacturer Zilog Corp. The Virtex-II Pro, Virtex-4, Virtex-5, and Virtex-6 FPGA families are particularly focused on system-on-chip (SOC) designers because they include up to two embedded IBM PowerPC cores The ISE Design Suite is the central electronic design automation (EDA) product family sold by Xilinx. The ISE Design Suite features include design entry and synthesis supporting Verilog or VHDL, place-and-route (PAR), completed verification and debug using Chip Scope Pro tools, and creation of the bit files that are used to configure the chip

RESULT FOR DELETE AND INSERT SORTING ALGORITHM

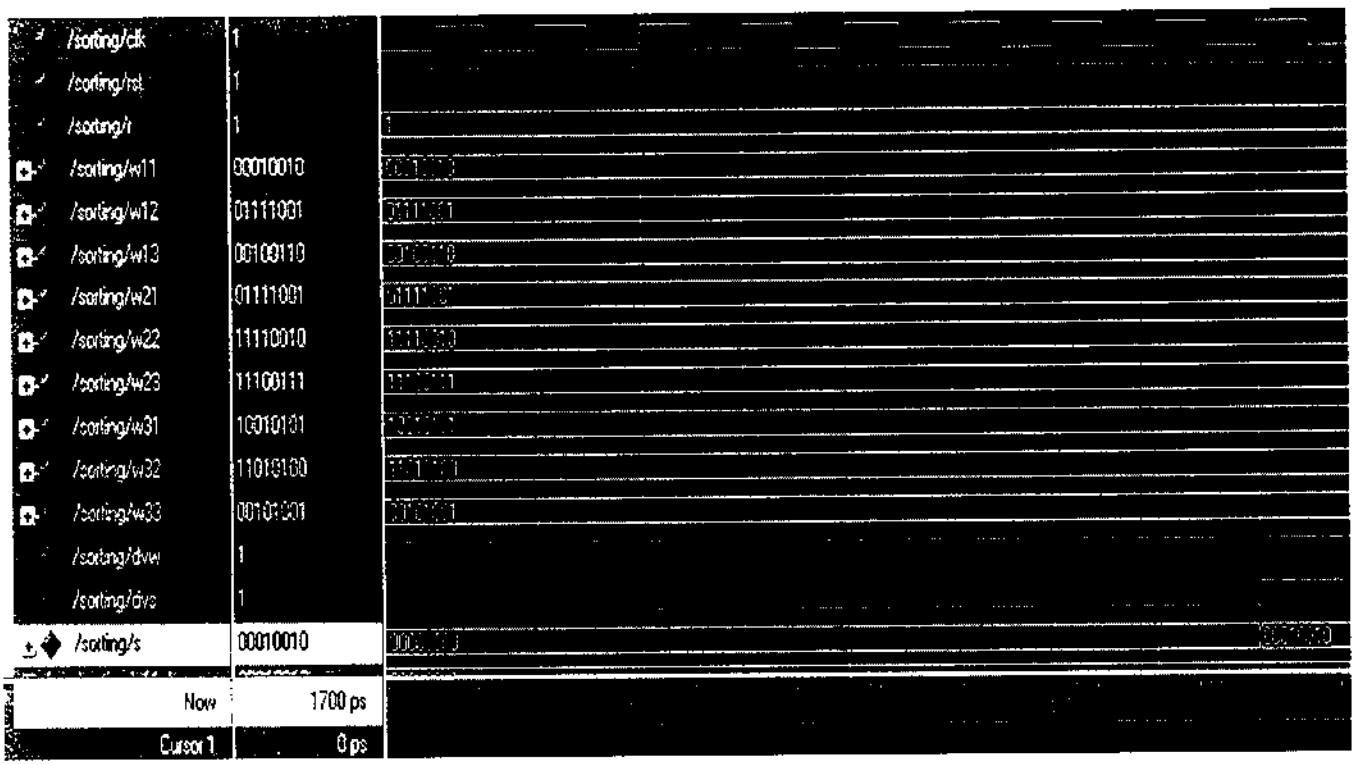

Fig 7.1 Simulation result for Delete and Insert sort algorithm

Input :

Clk= clk

Reset =0/1

Rank value

Output :

The values are sorted according to the rank value.

RESULT FOR ROF ALGORITHM

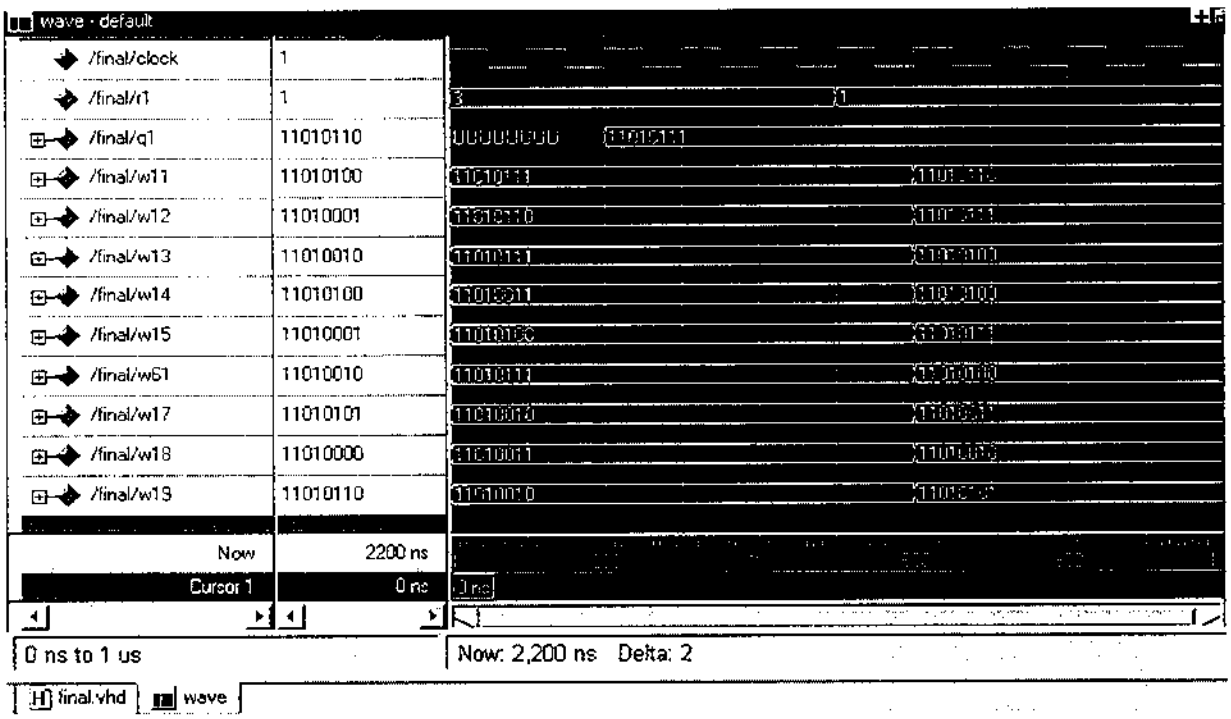

Fig 7.2 Simulation result for ROF algorithm

Input:

Reset=0/1

Clk=clk

R=1

Output :

The result are obtained according to the rank valuc

RESULT FOR LEVEL-QUANTIZER:

Case:1

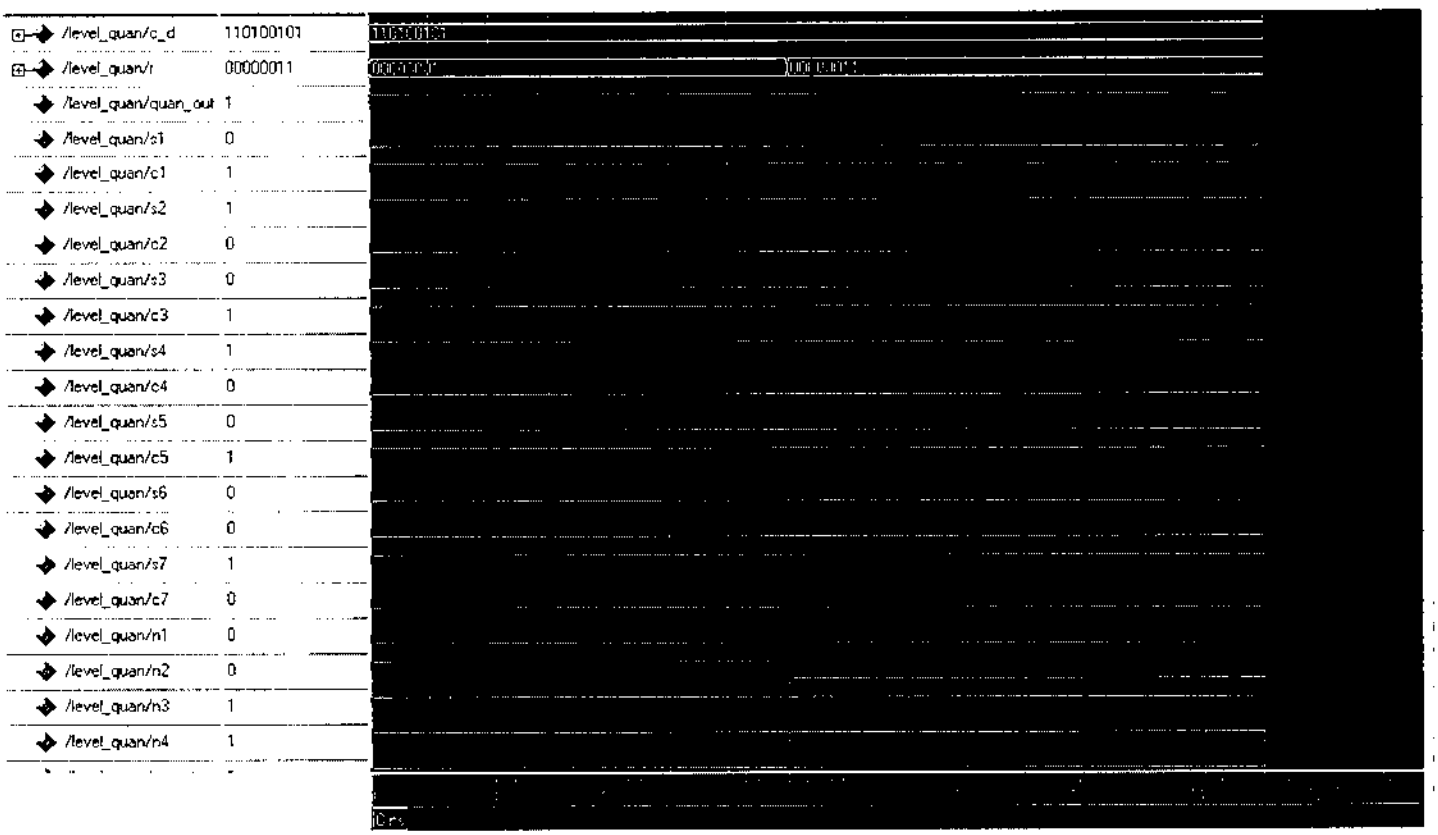

Fig 7.3 Simulation result for level-quantizer

Input: 110100101

Rank : 00000011

Output: 1

Case 2:

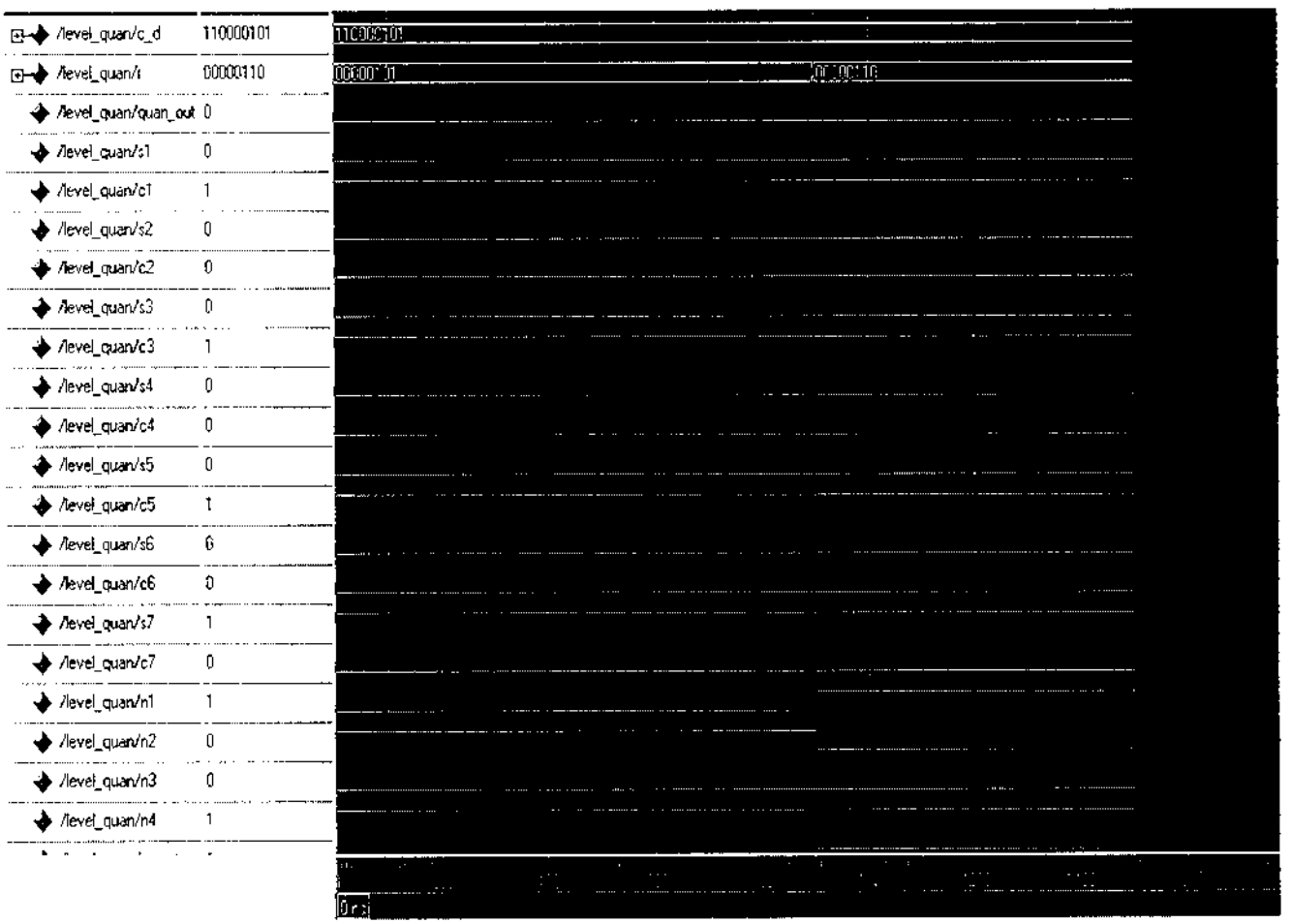

Fig 7.4 Simulation result for level quantizer

Input: 110000101

Rank: 0000110

Output: 0

RESULT FOR SHIFT REGISTER:

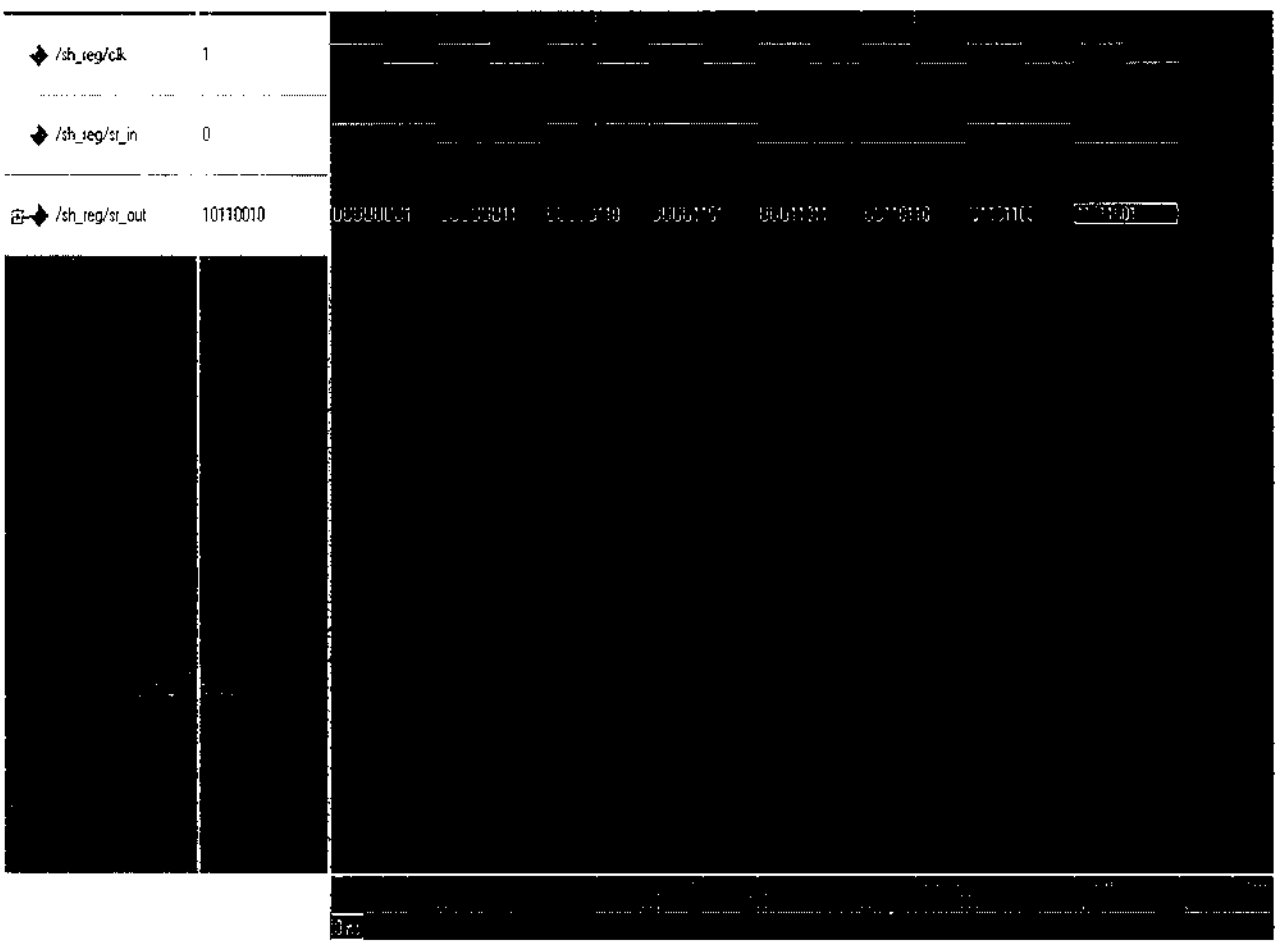

Fig 7.5 Simulation result for shift register

Input:

Clk = Clk

Output: The data obtained after polarization selector will be the output

RESULT FOR COMPUTING FIELD:

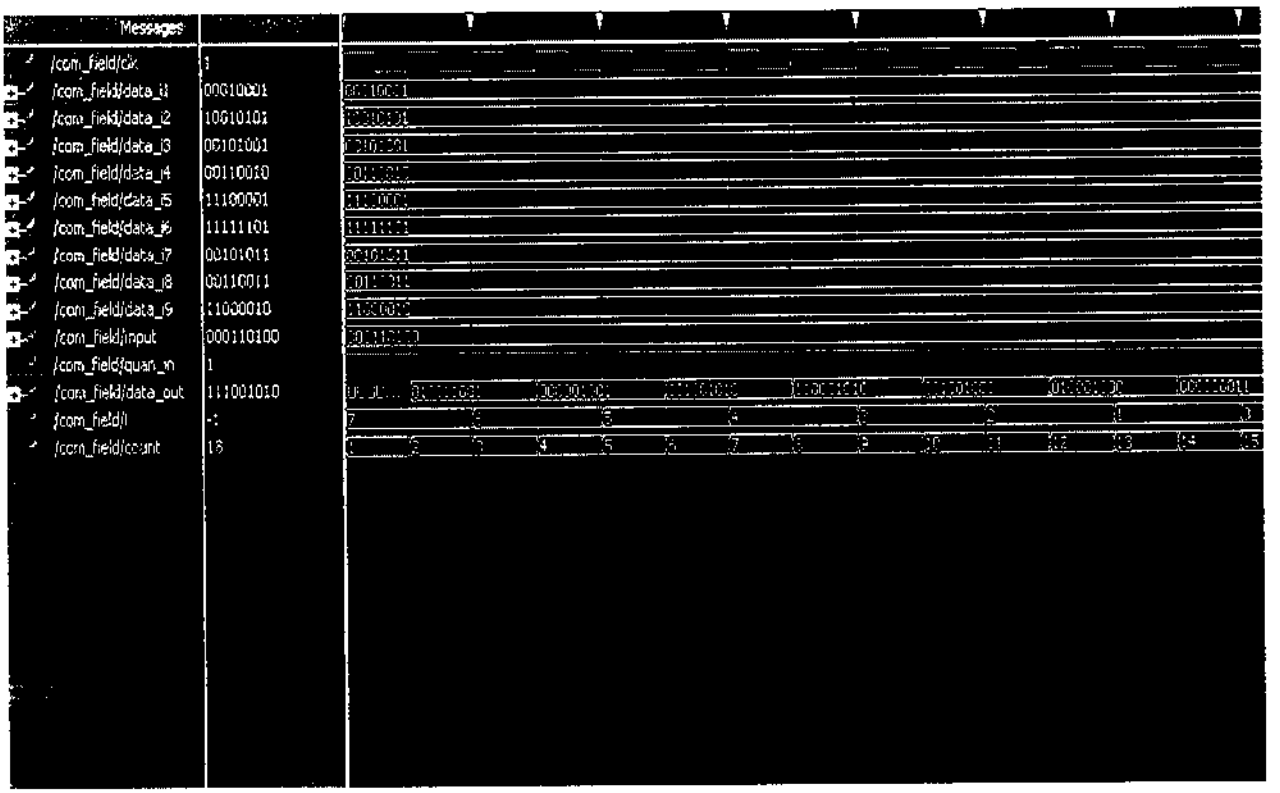

Fig.7.6. Simulation result for computing field

Input:

Clk= Clk

Output:

The data copied from datafield is taken for bit counting operations.

RESULT FOR POLARIZATION SELECTOR

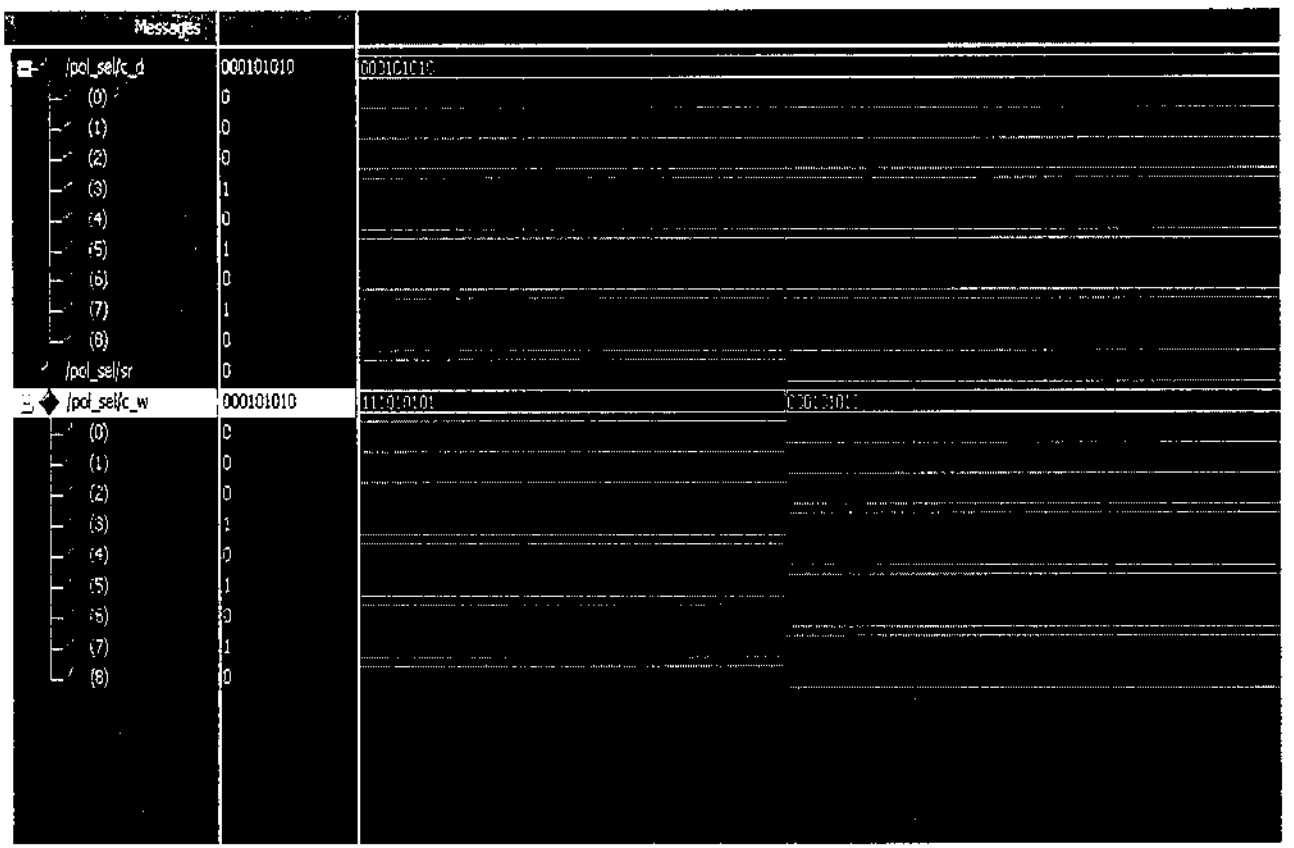

Fig.7.7. Simulation result for Polarization Selector

Input

c_d=000101010 which is to be polarized

Output: The polarized data

RESULT FOR DUALL CELL RANDOM ACESS MEMORY

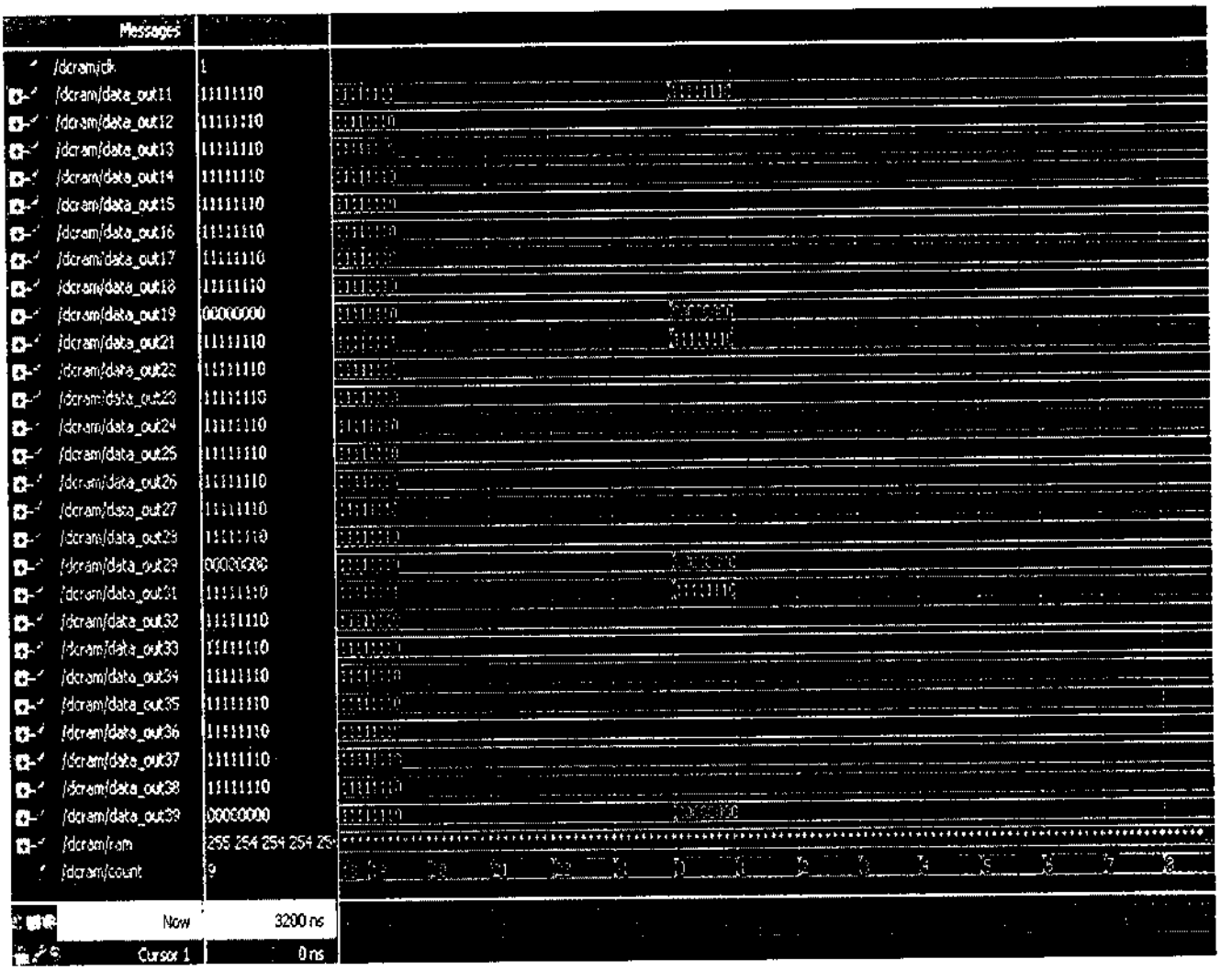

Fig.7.8. Simulation result for Dual Cell Random Access Memory

The input Image value is loaded to DCRAM

RESULT FOR RANK ORDER FILTER

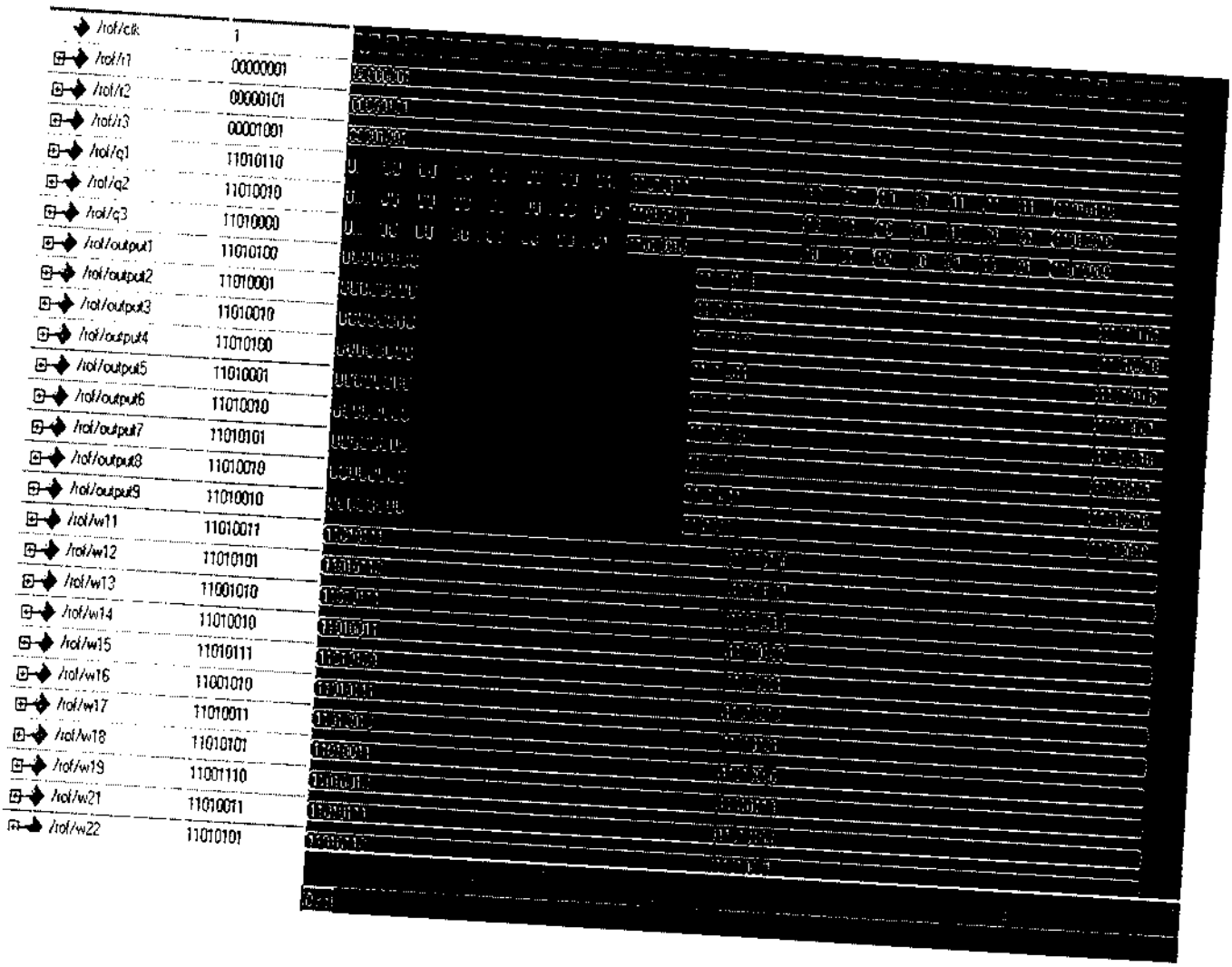

Fig.7.9. Simulation result for Rank Order Filter

Input:

Clk= clk

R1=1 R2=5 R3=9

Output:

The output is obtained according to rank and value of R1 and R3 get replaced by R3

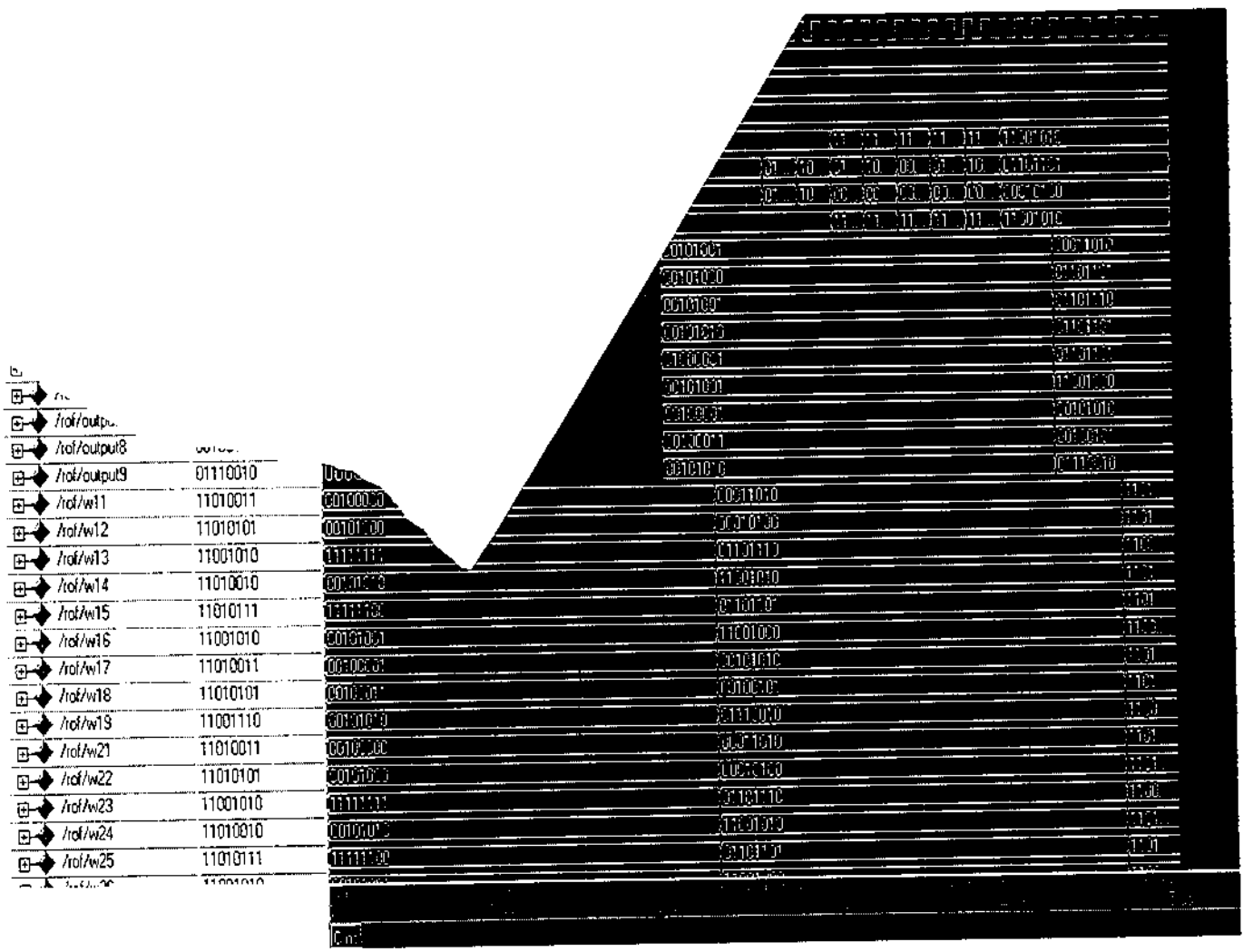

Fig.7.10. Simulation result for Adaptive Median Filter

Input:

Clk= clk

R1=1 R2=5 R3=9

Output: The median value of window is compared with neighborhood values and replaced the median value.

IMAGES TAKEN FOR ANALYSIS

Input:

ORIGINAL IMAGE

NOISE IMAGE

Output:

DCRAM ROF

AMF

Fig. 7.11 Image of size 256 x 256 taken for input and output

XILINX REPORT FOR SORTING ALGORITHM:

Design Overview for sorting

Property	Value
Project Name:	c:\sor
Target Device:	xc2s50e
Report Generated:	Sunday 04/03/11 at 08:10
Printable Summary (View as HTML)	sorting_summary.html

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slices:	691	768	89%	
Number of Slice Flip Flops:	1030	1536	67%	
Number of 4 input LUTs:	799	1536	52%	
Number of bonded IOBs:	116	182	63%	
Number of GCLKs:	1	4	25%	

Fig 7.12 Map report for sorting algorithm

Total equivalent gate count for design: **13,766**

Additional JTAG gate count for IOBs: 5,568

Peak Memory Usage: 105 MB

TIMING REPORT:

Minimum period: 7.693ns (**Maximum Frequency: 129.988MHz)**

Minimum input arrival time before clock: 25.771ns

Maximum output required time after clock: 6.613ns

Maximum combinational path delay: No path found

XILINX REPORT FOR ROF ALGORITHM

Design Overview for eg

Property	Value
Project Name:	c:\rofa
Target Device:	xc2s50e
Report Generated:	Saturday 04/02/11 at 23:51
Printable Summary (View as HTML)	eg_summary.html

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slices:	69	768	8%	
Number of Slice Flip Flops:	42	1536	2%	
Number of 4 input LUTs:	78	1536	5%	
Number of bonded IOBs:	113	182	62%	
Number of GCLKs:	1	4	25%	

Fig 7.13 Map report for rof algorithm

Total equivalent gate count for design: **1,074**

Additional JTAG gate count for IOBs: 2,400

Peak Memory Usage: 96 MB

TIMING REPORT:

Minimum period: 14.183ns (**Maximum Frequency: 70.507MHz**)

Minimum input arrival time before clock: 8.392ns

Maximum output required time after clock: 6.514ns

Maximum combinational path delay: No path found

XILINX REPORT FOR ROF USING DCRAM

Design Summary

Logic Utilization:

Number of Slice Flip Flops: 919 out of 13,824 6%
Number of 4 input LUTs: 1,946 out of 13,824 14%

Logic Distribution:

Number of occupied Slices: 1,270 out of 6,912 18%
Number of Slices containing only related logic: 1,270 out of 1,270 100%
Number of Slices containing unrelated logic: 0 out of 1,270 0%

Total Number 4 input LUTs: 2,457 out of 13,824 17%
Number used as logic: 1,946
Number used as a route-thru: 511
Number of bonded IOBs: 108 out of 510 21%
IOB Flip Flops: 72
Number of Block RAMs: 3 out of 72 4%
Number of GCLKs: 1 out of 4 25%
Number of GCLKIOBs: 1 out of 4 25%

Total equivalent gate count for design: **73,850**
Additional JTAG gate count for IOBs: 5,232

TIMING REPORT

Timing Summary:

Speed Grade: -7

Minimum period: 15.398ns **(Maximum Frequency: 64.943MHz)**

Minimum input arrival time before clock: 13.108ns

Maximum output required time after clock: 7.427ns

Maximum combinational path delay: No path found

XILINX REPORT FOR ADAPTIVE MEDIAN FILTER

MAP REPORT:

Design Summary

Logic Utilization:

Number of Slice Flip Flops: 1,131 out of 13,824 8%
Number of 4 input LUTs: 2,450 out of 13,824 17%

Logic Distribution:

Number of occupied Slices: 1,587 out of 6,912 22%
Number of Slices containing only related logic: 1,587 out of 1,587 100%
Number of Slices containing unrelated logic: 0 out of 1,587 0%

Total Number 4 input LUTs: 3,074 out of 13,824 22%
Number used as logic: 2,450
Number used as a route-thru: 624

Number of bonded IOBs: 116 out of 510 22%
IOB Flip Flops: 72
Number of Block RAMs: 3 out of 72 4%
Number of GCLKs: 1 out of 4 25%
Number of GCLKIOBs: 1 out of 4 25%

Total equivalent gate count for design: **79,398**
Additional JTAG gate count for IOBs: 5,616

TIMING REPORT

Timing Summary:

Minimum period: 22.304ns (**Maximum Frequency: 44.835MHz**)

Minimum input arrival time before clock: 14.406ns

Maximum output required time after clock: 7.517ns

Maximum combinational path delay: No path found

FLOOR PLANNING FOR SORTING ALGORITHM

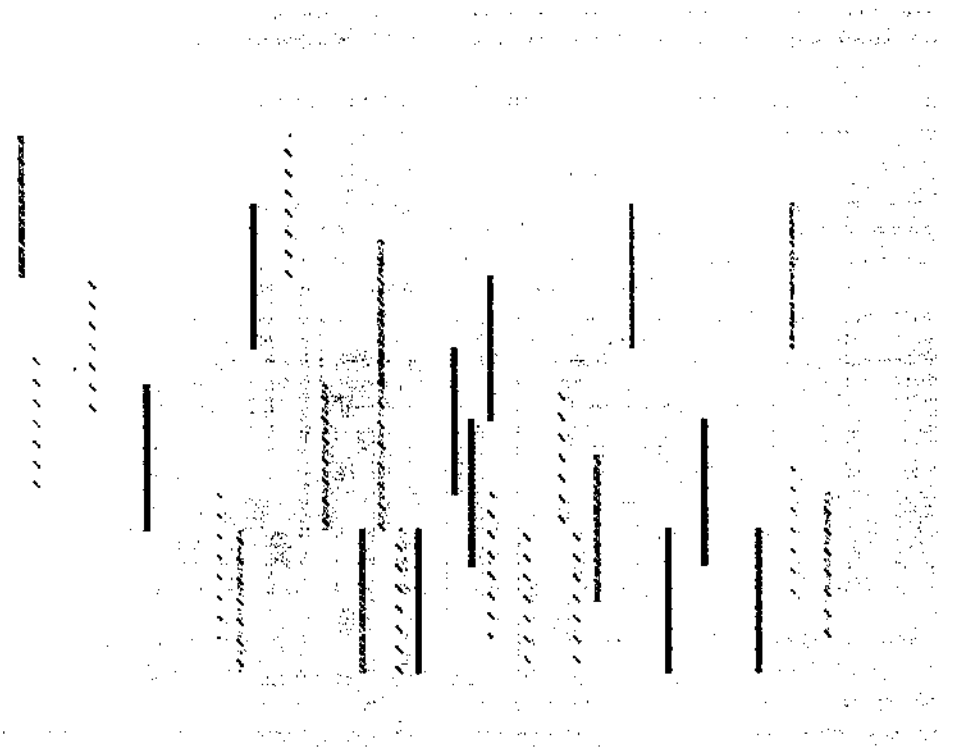

Fig 7.14 Floor planning for sorting algorithm

PIN LOCATIONS FOR SORTING ALGORITHM

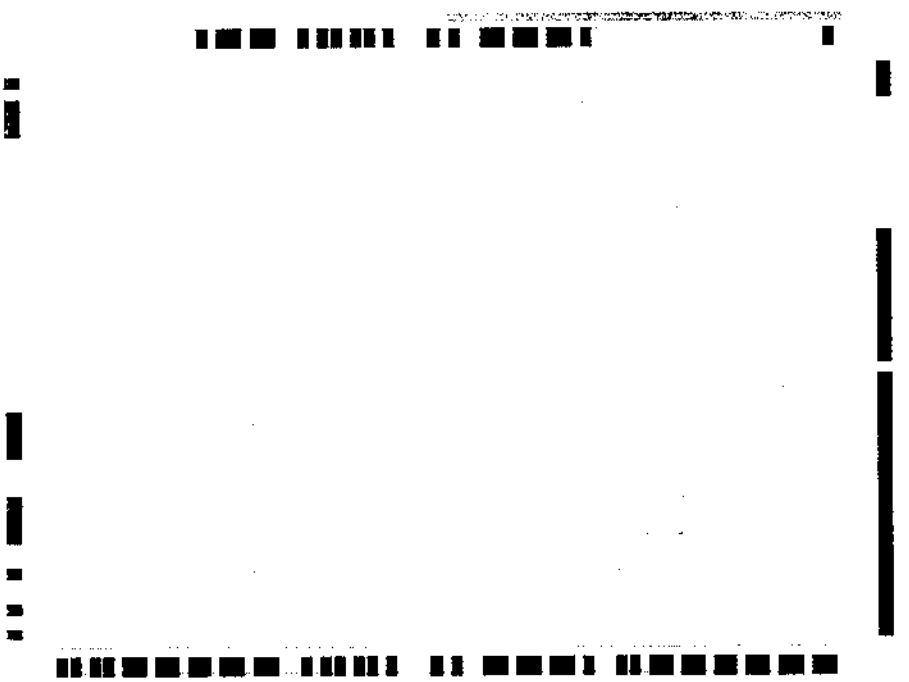

Fig 7.15 Pin location for sorting algorithm

FLOOR PLANNING FOR ROF ALGORITHM

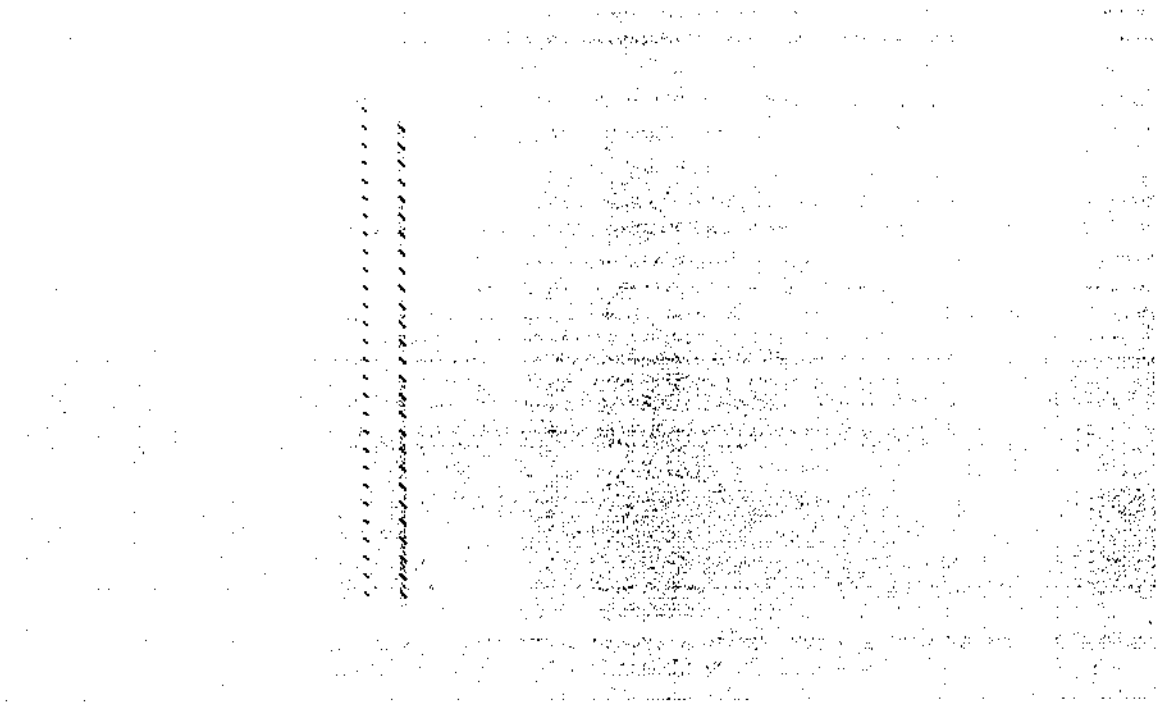

Fig 7.16 Floor planning for ROF algorithm

PIN LOCATIONS FOR ROF ALGORITHM

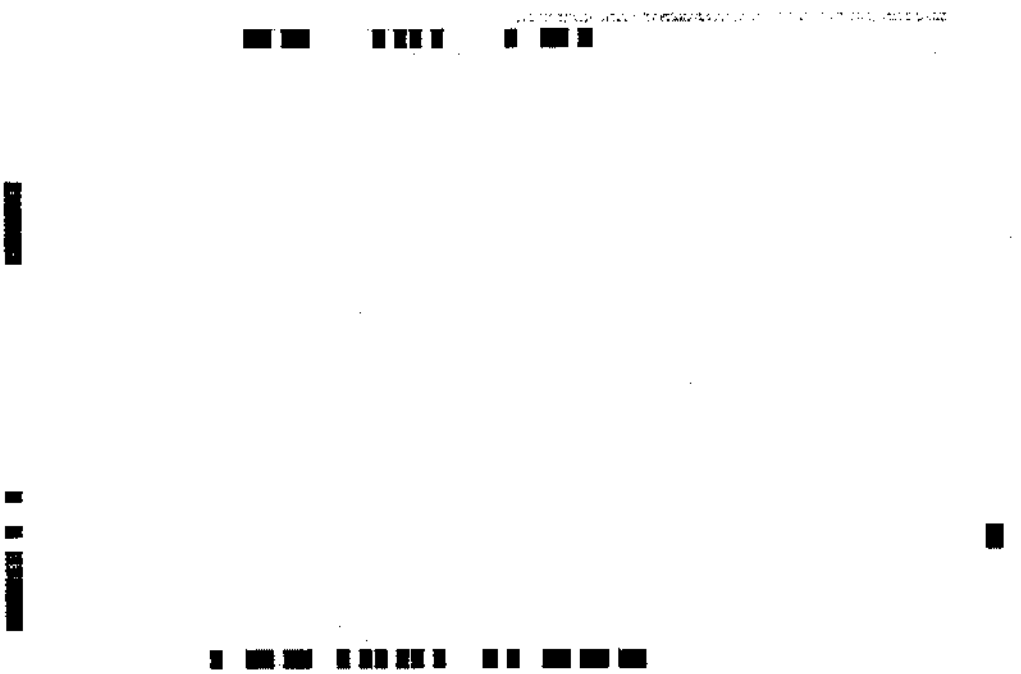

Fig 7.17 Pin location for ROF algorithm

CHAPTER-7
CONCLUSION

When the ROF algorithm is compared with sorting algorithm it produces result with reduction in **gate count utilization.** The sorting algorithm uses total gate count for designing is **13,766** with maximum frequency of **129.988MHz** whereas for ROF algorithm uses total gate counts for designing is **1074** with maximum frequency of **70.507MHz .**

The ROF algorithm is implemented using **DCRAM** architecture. The architecture is pipelined which processes one pixel per clock cycle, thus to process an image of size *256* x 256 it requires maximum frequency used is **64.943MHz** and hence is suitable for real time applications The **adaptive median filter** successfully removes impulsive noise from images at clock frequency of **44.835MHz**. It does a reasonably good job of smoothening images that contain impulsive noise better than the Rank Order Filter.

FUTURE WORK :

The future work of this project is to do a FPGA implementation and replace the median filter in Adaptive Median Filter design with ROF.

BIBILIOGRAPHY

[1] R.Roncella, R.Saletti, P.Terreni, "70-MHz 2-μm CMOS Bit-level Systolic Array Median Filter," *JSSC*, vol. 28, pp. 530-536, May 1993.

[2] C.Chakrabarti, "Sorting Network Based Architectures for Median Filters," *IEEE Trans. Circuits and Systems-II: Analo'g Digital Signal Proc.*, vol. 40, pp. 723-727, Nov. 1993.

[3] T.S.Huang, G.J.Yang, G.Y.Tang, "A Fast Two- Dimensional Median Filtering Algorithm," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-27, pp. 13-18, Feb. 1979.

[4] L.W.Chang, S.S.Yu, "A New Implementation of Generalized Order Statistic Filter by Threshold Decomposition," *IEEE Trans. Signal Proc.*, vol. 40, pp. 3062-3065, Dec. 1992.

[5] C.Chakrabarti, "High Sample Rate Array Architectures for Median Filters," *IEEE Trans. Signal Proc.*, vol. 42, pp. 707-712, Mar. 1994.

[6] B.K.Kar, D.K.Pradhan, "A New Algorithm for Order Statistic and Sorting", *IEEE Trans. Signal Proc.*, vol. 41, pp. 2688-2694, Aug. 1993.

[7] K.Chen, "Realizations of a Class of Non-Linear Filters Using a Bit-Serial Approach," ISCAS '88, pp. 1749-1752.

[8] L.W.Chang, J.H.Lin, "A Bit-Level Systolic Array for Median Filter", *IEEE Trans. Signal Proc.*, vol. 40, pp. 2079-2083, Aug. 1992.

[9] C.L.Lee, C.W.Jen, "Binary Partition Algorithms and VLSI Architectures for Median and Rank Order Filtering," *IEEE Trans. Signal Proc.*, vol. 41, pp. 2937-2942, Sep. 1993.

[10] J.P.Fitch, "Software and VLSI Algorithms for Generalized Ranked Order Filtering," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 553-559, May 1987. [37] [11]V.A. Pedroni, Compact hamming-comparator-based rank order filter for digital VLSI and FPGA implementations, in: IEEE International Symposium on Circuits and Systems, vol. 2, May 2004, pp. 585–588.

[12]. S. Singh, S. Azmi, N. Agrawal, P. Phani, A. Rout, Architecture and design of a high performance SRAM for SOC design, IEEE International Symposium on VLSI Design, January 2002, pp. 447–451.

[13] T.S.Huang, G.J.Yang, G.Y.Tang, "A Fast Two-Dimensional Median Filtering Algorithm." *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-27, pp. 13-18, Feb. 1979.

[14] L.W.Chang, S.S.Yu, "A New Implementation of Generalized Order Statistic Filter by Threshold Decomposition," *IEEE Trans. Signal Proc.*, vol. 40, pp. 3062-3065, Dec. 1992.

[15] C.Chakrabarti, "High Sample Rate Array Architectures for Median Filters," *IEEE Trans. Signal Proc.*, vol. 42, pp. 707-712, Mar. 1994.

[16] B.K.Kar, D.K.Pradhan, "A New Algorithm for Order Statistic and Sorting", *IEEE Trans. Signal Proc.*, vol. 41, pp. 2688-2694, Aug. 1993.

[17] K.Chen, "Realizations of a Class of Non-Linear Filters Using a Bit-Serial Approach," ISCAS '88, pp. 1749-1752.

[18] L.W.Chang, J.H.Lin, "A Bit-Level Systolic Array for Median Filter", *IEEE Trans. Signal Proc.*, vol. 40, pp. 2079-2083, Aug. 1992.

[19] C.L.Lee, C.W.Jen, "Binary Partition Algorithms and VLSI Architectures for Median and Rank Order Filtering," *IEEE Trans. Signal Proc.*, vol. 41, pp. 2937-2942, Sep. 1993.

[20] J.P.Fitch, "Software and VLSI Algorithms for Generalized Ranked Order Filtering," *IEEE Trans.Circuits Syst.*, vol. CAS-34, pp. 553-559, May 1987.