Design and Implementation of Stochastic Architecture for Edge Detection

THESIS SUBMITTED

IN FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

MASTER OF ENGINEERING

IN

ELECTRONICS & TELECOMMUNICATION ENGINEERING

By

MALAY MANDAL

EXAM ROLL- M4ETC19016

REGISTRATION NO- 140699 of 2017-2018

Under the guidance of

Prof. MRINAL KANTI NASKAR

Department of Electronics & Telecommunication Engineering

Jadavpur University, Kolkata- 700032

West Bengal, India

May, 2019

Faculty of Engineering and Technology

Jadavpur University

This is to certify that the thesis entitled —"Design and Implementation of Stochastic Architecture for Edge Detection" has been carried out by MALAY MANDAL (Class Roll No: 001710702014, Examination Roll No.: M4ETC19016 and Registration No: 140699 of 2017-2018) under my guidance and supervision and be accepted in partial fulfilment of the requirement for the degree of Master of Electronics & Telecommunication Engineering.

____

Prof. Mrinal Kanti Naskar

Supervisor

Department of Electronics & Telecommunication Engineering

Jadavpur University

Kolkata-700032

____

Prof . Sheli Sinha Chaudhuri

Head of the Department

Electronics & Telecommunication Engineering

Jadavpur University

Kolkata- 700032

____

Prof . Chiranjib Bhattacharjee

Dean

Faculty council of Engineering & Technology

Jadavpur University

Kolkata- 700032

Faculty of Engineering and Technology

Jadavpur University

CERTIFICATE OF APPROVAL

The forgoing thesis titled —"Design and Implementation of Stochastic Architecture for Edge Detection" is here by approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn there in but approve the thesis only for the purpose for which it has been submitted.

Committee on final Examination	___________________________
for evaluation of the Thesis:	**Additional Examiner**

____

Supervisor

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original work by the undersigned candidate, as part of his Master of Electronics and Telecommunication studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name	:	MALAY MANDAL
Exam roll	:	M4ETC19016
Title	:	”Design and Implementation of Stochastic Architecture for Edge Detection”

____

Signature of candidate

Acknowledgment

The success and the final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project.

I respect and thanks to my project guide Prof. Mrinal Kanti Maskar, for providing me an opportunity to do the project work and giving me all support and guidance which made me complete the project duly. I am extremely thankful to him for providing such a nice support and guidance, although he had busy schedule managing the corporate affairs.

I am also thankful and fortunate enough to constant encouragement, support and guidance from all the teachers of the Department of Electronics and Telecommunication Engineering, who taught me during the first year of my course work. I would like to express my thanks to my senior Mr. Mithun Kumar Mahato whose thesis work helped me to get a quick start in my project. I also very thankful to Ms. Sumana Biswas, Dr. Rajarshi Middya, Ms. Shyamali Mitra and all other lab mates for their suggestions and help during the entire phase of this project work.

I owe my deepest gratitude to my parents, siblings, friends for their constant encouragement and support, which keep me interested on my focus and they guided me all along, till the completion of my work.

MALAY MANDAL

Abstract

In the recent time stochastic computing, operates on random bit streams which is representing the value of probability, to implement digital computational element. To represent the probability value x by a bit stream, then $x * 100\%$ bit of that bit stream must be $1's$. Using this simple concept, stochastic computing can be used to perform complicated work with much less hardware as compared to conventional binary methods: e.g., only a AND gate can be used to implement multiplication between two bit-streams which are uncorrelated.
Stochastic computing (SC) was introduced in the 1960s as a low-cost alternative to conventional binary computing. This is a unique method as probability values represents the process information and every computation value. Stochastic computing requires very simple arithmetic units which was a preliminary design concern in the past. Although it has above mentioned advantage and high capability of error tolerance, stochastic computing was not as popular as conventional binary method because of its little bit low accuracy and long computation times. This thesis is about the application of stochastic computing in a field where the small size, error tolerance, and probabilistic features of SC may be compared to the conventional methodologies successfully. First, we will take some key concepts of stochastic number representation and various stochastic combinational and sequential. In this thesis, we first introduce the stochastic tools to compute any given function in general and then apply these tools for various applications. We give a brief survey of the stochastic implementations of various image processing applications and observe that all the image processing tasks using stochastic computing. Finally, we will give examples of the potential applications of SC and discuss some practical problems that are yet to be solved.

Contents

List of figure

List of Tables

Chapter 1

Introduction

Introduction

1.1. Stochastic computing

In 1960s, stochastic computing was introduced firstly as a combination of digital-analog computing technique that gives designers a confidence to implement complex arithmetic function using very less no of hardware. But later transistor became very cheaper and designers were attracted to the conventional system due to its fast response, even conventional method is less effective in chip area and cost. Due to this stochastic computing had very few application area. But now after development of technology, new challenge is to reduce the hardware, chip area and energy consumption for the circuit. The stochastic computing system can be used to solve this type of problem faced by the conventional system with the high accuracy.

Stochastic computing is a re-emerging computing technique that was invented in the 1960s as a low-cost alternative to conventional binary computing [1]. It is unique in that it represents and processes information in the form of probabilities. More recently, it has been shown to be useful in important applications such as image-processing and communications. In this dissertation, we address some of the challenges of SC, and show how it can be used to implement efficient image-processing circuits. This chapter provides the motivation behind this work, as well as a brief introduction to SC and a summary of our contributions.

Stochastic computing is going to be more popularized because of low cost in terms of hardware area, high fault-tolerance and lesser complexity compared to computations using conventional binary method. A very simple logic can compute all the relevant computation by the stochastic computation method

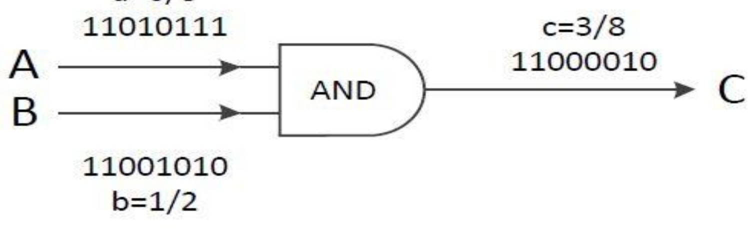


fig 1.1: multiplication can be done by using a single AND gate

Combinational logic were implemented in the earlier stochastic computing. For example, only an AND gate can behave like a multiplier as shown in the Figure 1.1, but the input must be uncorrelated, and can calculate the multiplication with high accuracy with the low fault- tolerance and lower noise sensitivity.

Stringent application requirements such as smaller size and lower power consumption with high reliability are continuously challenging the modern computing hardware. Moreover, future device technologies such as nanoscale CMOS transistors are expected to become ever more sensitive to system and environmental noise and to process, voltage, and thermal variations [2]. When we are trying to design a system with the conventional design method then our main focus or target is to get an error free results and very fast response to the input. For instance, to reduce the effect of the noise in circuit level, the operating voltage and the size of the transistor should be high in the critical circuit, but as a result the packaging density will be low as hardware area will be higher. At the architecture level, fault-tolerant techniques such as triple modular redundancy (TMR) can further enhance system reliability. The trade-off for the fault tolerance is typically more hardware resources. As device scaling continues, this overdesign methodology will consume even more hardware resources, which can limit the advantages of further scaling [2]. All of these issues can be solved efficiently with the unconventional methodologies with little bit approximation and stochastic computing is one of them.

In stochastic computing, any number is normalized to a value which is belongs to $[0,1]$ or we can say any number is mapped to a probability value which can be represented by a stream of bits. For example, if a bit-stream is containing 20 percent $1's$ and 80 percent $0's$ denotes the stochastic number X with value $p_x = .20$, gives the concept that the probability of occurrence of 1's in the bit-stream position is X. bit stream may have any length. Neither the length nor the structure of X need be fixed; so bit-streams 1000, 0100 and 0100, 0100 are all possible representations of 0.25. Note that the value p_x depends on the ratio of no of $1's$ to the length of the bit-stream, not on their positions, $1's$ can choose any position of the bit streams randomly. This bit stream can be generated by stochastic number generator, will be discussed later.

For computation stochastic computation needs significantly less hardware compared to conventional binary methodology. However, this stochastic method suffer from significant increase in computation time because a number is represented by large no of bits, called bit stream. For example, 1024 levels can be represented by 10 bit binary no. For the same resolution, a stochastic number should be represented using 1024 bits. The stochastic computing systems are ideal for low-speed applications such as neural networks [3], cyber-physical systems operating at very low rates, and biomedical applications. By stochastic computing logic we cannot get exact value, but we can get approximate value which is very closer to the exact value. In many applications, such as machine learning, where a decision is made based on a thresholding operation, the decisions may not be affected by approximate computing. Stochastic computing is also exploited for decoding of error-correcting codes (ECC) [4] [5]. Another main advantage of stochastic computing is its inherent tolerance to faults in CMOS circuits [6].

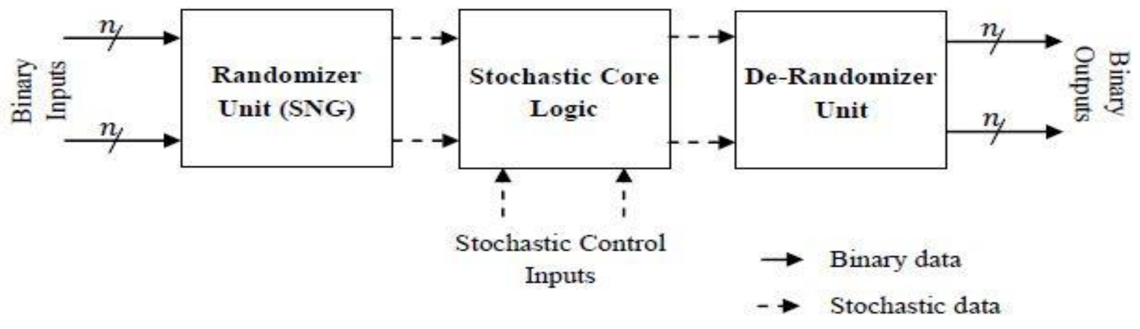


fig 1.2: block diagram of stochastic computing

Another advantage of stochastic computing, stochastic computing circuits are fault-tolerant. The value of a stochastic number X is only depends on the no of $1's$ in the bit-stream, and not where they are positioned. This makes the representation highly redundant, and hence fault-tolerant. As an example, consider the bit-stream 01001010 which is an SN of value $\frac{3}{8}$. In a noisy environment, the bits are susceptible to soft errors of the bit-flip type [7]. Should a bit-flip occur in the stochastic computing, its value will slightly change to $\frac{4}{8}$ (or $\frac{2}{8}$). Multiple bit-flips may even cancel each other out. On the other hand, bit-flips can greatly alter the value of a conventional binary number, if they occur on the most significant bits. For instance, if we consider the same number $\frac{3}{8}$ in conventional binary format 0.011, a single bit-flip causes a huge error if it affects a high-order bit. A change from 0.011 to 0.111, for example, changes the result from $\frac{3}{8}$ to $\frac{7}{8}$. Stochastic number does not have any higher significant bit or lower significant bit like conventional binary methodology, in stochastic number all bits are having equal significance.

1.2. Edge Detection

The focus of this thesis is the implementation of edge detector using the stochastic computing. Edge detection is basically the segmentation method which is based on the sharp change of intensity in an image. There are three types of image features in which we are interested are isolated points, lines and edges. Edge pixels are that pixels where the intensity of the images are changing abruptly from one region to another region. Basically edge pixels divide an image to different regions, and for which we can identify the objects. For an image if there is an object with the background then by getting the edge of the object we can get the nature and other information of that image. For which we need less storage capacity to store the information of the image. Edge detectors are local image processing methods designed to detect edge pixels. A line may be viewed as an edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels [8]. We can detect abrupt changes in intensity using derivatives. First order and second order derivatives are more suitable for the edge detection purpose. We require that any approximation used for the first order derivative (i) must be zero in areas of constant intensity; (ii) must be non-zero at the onset of an intensity step or ramp; and (iii) must be non-zero at points along an intensity ramp, similarly for a second order derivative required approximations are (i) must be zero in areas of constant intensity; (ii) must be non-zero at the onset and end of an intensity step or ramp; and (iii) must be zero along intensity ramp [8]. Because we are working with finite digital quantities, so maximum value changes in the intensity level will be also a finite value and the shortest distance over in that a change of intensity can be occurred between the adjacent pixels.

The points at which image brightness changes abruptly are typically organized into a set of curved line segments termed as edges. Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction [9].

The main purpose of edge detection is to eliminate the unnecessary information of the image while preserving the structure of the image, extraction of important features of the image like corners, curves, lines, and recognition of objects, boundaries, segmentation. Edge detection is part of the computer vision and recognition. There are many types of edges like roof edge, step edge, ramp edge, line edge. But real edges are noisy and discrete.

So we want an edge detector that produces (i) edge magnitude (ii) edge orientation (iii) high detection rate and (iv) high localization.

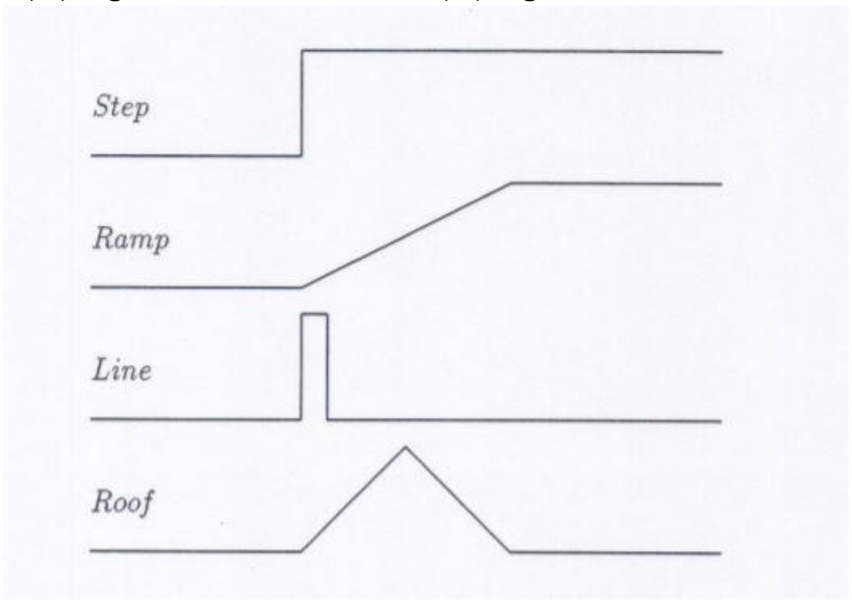


fig 1.3: one dimensional edge type

There are many types of edge detector. Few of them are classical gradient edge detector, Gaussian based filters, wavelets used for different scales, fuzzy logic and neural networks etc. Roberts, prewitt, sobel, kirsch and Robison are the examples of classical gradient edge detector. Marr and hildreth, canny, shunck, witkin and Bergholm are examples of Gaussian based filters. Laplacian mask is the example of second order edge detector.

During the edge detection first we do smoothing. In this section noise reduction can be done by the smoothing. Then we follow the step called enhancement, by enhancement edges are going to be more sharped. Then we do detection, in this case we decide which one is to be discarded and which one is to be maintained by thresholding process. Then localization, that is determination of exact location of the image, edge thinning and linking are usually required in this step.

The output set of edges in the output image by the edge detector can be classified in two categories, (i) correct edge (ii) false edge. Correct edges are those edges which are detected by the edge detector and also present in the original image. False edges can be further classified in two categories (a) false positive and (b) false negative. False positive edges are those edges which is detected by the edge detector but those are not present in the original image, they are called false edge. False negative are those edges which are present in the original image but edge detector can't detect it, they are called missing edge.

1.3. Previous work

In previous work, Gaines [1] introduced stochastic computing systems. After that Brown and Card [3] presented stochastic computational elements for neural computation, Qian et al[10] proposed architecture for synthesis of stochastic circuits and polynomial computation using combinational logic, Li et al[11][12][13][14] presented the mathematical analysis of these elements including sequential logic synthesis and also presented applications of SC in image processing; Najafi et al. [15, 16] showed high-speed stochastic circuits using synchronous analog pulses and also showed the skew-tolerance behavior of stochastic circuits. Based on these works we present a brief survey on the image processing applications of stochastic computing.

1.4. Organization of the thesis

This thesis is organised as follows. In chapter two, we will discuss the basic concept of the edge detection algorithm. In that section we will learn about the various method of the edge detection and their working principle. In the chapter three we will discuss the Stochastic Architecture of the Edge Detection and how edge detector can be implemented by the stochastic logic circuit. In chapter four we will learn about the realization of the stochastic edge detector. In this section we will discuss about the various stochastic element which are used in the implementation of the edge detector. In the chapter five, we will see the results of the edge detection using stochastic logic and we will analyse the results. Finally we will conclude the thesis in the chapter six and we will discuss about the future scope.

Chapter 2
Edge Detection Algorithm

Edge Detection Algorithm

The initial stage of image processing identify the distinctive attributes in images that are closely connected to evaluate the structure and features of objects in an image. Edge detection is one of those aspect. Edges are formed where unexpected changes of intensity is observed and edges have a massive importance for estimating the images and to study the applicable information from the images. In most cases Edges are produced on the boundary of two regions in an image where from the edges to both side intensity is either very higher value or lower value. In common man languages, the edges actually divide image in two region. Edge detection is the restoring tool to the recovering of information from an image, which is a reason for edge detection to be one of the favoured area of research. Here, we are supposed to highlights the basic algorithms for edge detection.
We can find numerous algorithm for implementation of edge detection. Among them Robert's cross operator, Sobel, Prewitt, canny and Laplacian operator are some well known methods for the edge detection. Robert's cross operators are possibly from some foremost methods using 2-D masks with a diagonal preference. The Roberts operators are built on performing the diagonal differences. Excluding Laplacian operator, all the above mentioned operators are first derivative operators. The Laplacian operator is only the second derivative operator. From the early period of time Edge detection has been one of the mostly enjoyed research areas in computer vision. Two commonly used edge detectors were gifted by Roberts and Sobel in early period of computer vision. Prewitt, Hueckel, and Frei and Chen also presented few famous and regularly used edge detectors in early days. Many statistical and several filtering approaches have also been used for edge detection. In 1980's the widely used Edge detector Algorithms were based on the Laplacian of Gaussian and the gradient of Gaussian. In recent days, the Laplacian of Gaussian edge detection method is a dominant models of biological edge detection. Haralick gave an edge detection scheme depending on the theory of directional derivatives. His method assimilated the configuration of image smoothing based on estimating the image with local surface patches. Canny edge detection is also a widely discussed scheme of edge detection.

2.1. Robert’s cross operator

2.1.1. Brief description

The Roberts cross operator is very useful in image processing for edge detection. In 1963 Lawrence Robert proposed this edge detection algorithm. Being one of the simplest edge detection algorithm, this operator helps in a simpler, quicker to compute, 2-D spatial gradient measurement on an image. It thus focuses regions of high spatial gradient which often correspond to edges. In its most common usage, the input to the operator is a greyscale image, as is the output. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.

2.1.2. How it works

It finds out the unexpected changes in the intensity in the x and y direction [17], by using two 2×2 matrices G_x and G_y as shown in the figure 2.1. These two matrices are convoluted with the original images.

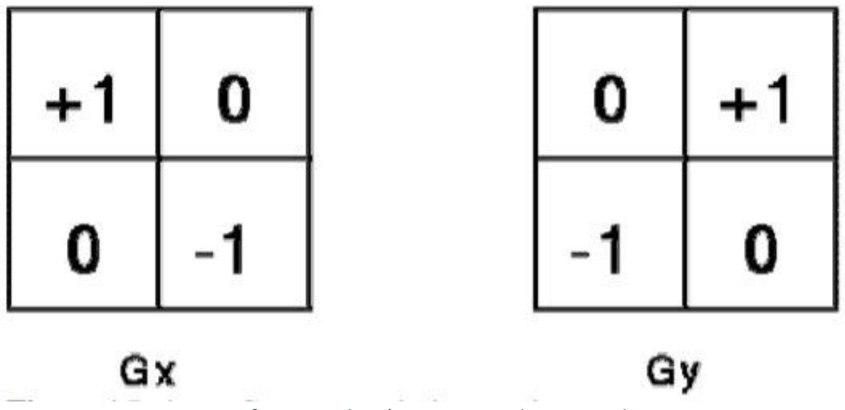


fig 2.1: Robert's cross convolution mask

These masks are arranged to respond maximally to edges running at $45°$ to the pixel grid, one mask for each of the two perpendicular orientations. The masks can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these G_x and G_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. To determine whether the pixel being evaluated is an edge pixel or not, the magnitude of the gradient is calculated as follows:

$$|G| = ({G_x}^2 + {G_y}^2)^{\frac{1}{2}}$$

Although typically, an approximate magnitude is computed using:

$$|G| = |G_x| + |G_y|$$

Using this we can compute more quicker.
If the calculated magnitude is greater than a minimum threshold value (set based on the nature and quality of the image being processed), the pixel is considered to be part of an edge.
The angle of orientation of the edge representing the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

In this case, orientation 0 represents the case where the direction of maximum contrast from black to white runs from left to right on the image, and other angles are measured anticlockwise with respect to this.
Often, the absolute magnitude is the only output the user sees --- the two components of the gradient are conveniently computed and added in a single pass over the input image using the pseudo-convolution operator shown in Figure 2.2.

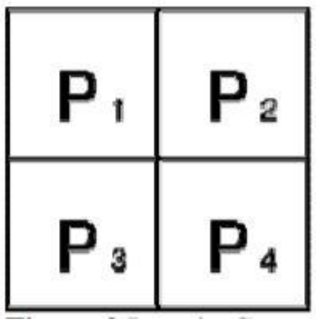


fig 2.2: pseudo convolution Robert's mask for quick computation

Using this mask the approximate magnitude is given by:

$$|G| = |(P_1 - P_4)| + |(P_2 - P_3)|$$

The small size of the masks for the Roberts Operator is helpful in implementation and quick to calculate the mask responses. However, these responses are also very sensitive to noise in the image.
Let $x_{i,j}$ denotes the pixel value at location (i,j) of the original image and $y_{i,j}$ is the pixel intensity value at location (i,j). the convolution between the original image and the Robert's cross operator and produce output as given as below:

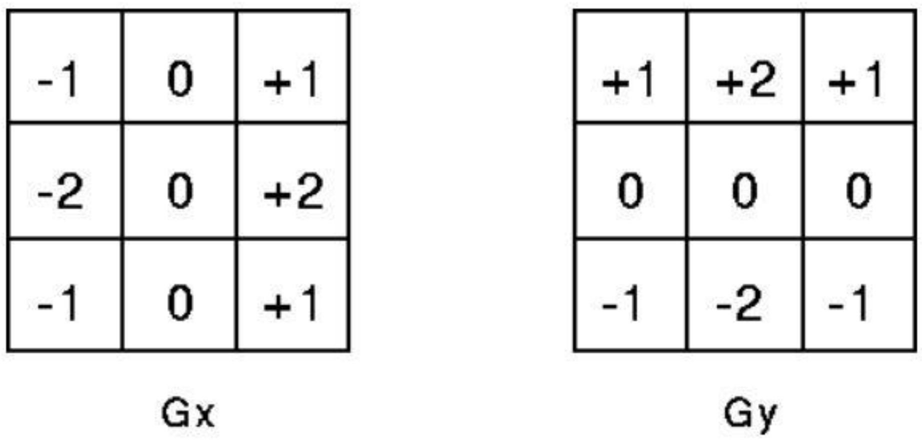
$$y_{i,j} = \frac{1}{2}\left[\, \left|x_{i+1,j+1} - x_{i,j}\right| + \left|x_{i,j+1} - x_{i+1,j}\right| \,\right]$$

2.2. Sobel edge detector

2.2.1. Brief description

The Sobel operator is useful to compute a 2-D spatial gradient measurement on an image. Hence, it focuses regions of high spatial gradient which corresponds to edges. More specifically it is used to commute an estimated absolute gradient magnitude at each point in an input greyscale image.

The Sobel Operator is a very usual operator to approximate the derivative of an image. It is distinct in the y and x orientations. Considering at the x-direction, we can see the gradient value of an image in the x-direction is same as this operator here. Here, We use the kernel 3 by 3 matrix, both for x and y orientation separately. In 3×3 kernel matrix for the gradient for x-direction negative numbers are kept on the leftmost column and positive numbers are kept on the right most column whereas the central pixel part is preserved. While calculating the gradient for y-direction we follow the same pattern but in rows i.e. negative numbers are kept on the bottom and positive numbers are on the top. Here also the central rows pixels are preserved.



-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

fig 2.3: Sobel operator for edge detection

2.2.2. How it works

This operator contains a pair of 3×3 convolution masks as shown in Figure 2.3. One mask is simply the rotation of prior one by $90°$. This is very similar to the Roberts Cross operator.

These masks are prepared to respond maximally to edges running vertically and horizontally relative to the pixel grid, one mask for each of the two perpendicular directions. For producing distinct measurements for the gradient of each directions, the masks are also used distinctly to the input image, (call these G_x and G_y). These can then be considered together to find the absolute magnitude and the orientation of the gradient at each point. The gradient magnitude is given by:

$$|G| = ({G_x}^2 + {G_y}^2)^{\frac{1}{2}}$$

Although typically, an approximate magnitude is computed using:

$$|G| = |G_x| + |G_y|$$

Using this we can compute more faster.

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

In this case, orientation 0 is taken to mean that the direction of maximum contrast from black to white runs from left to right on the image, and other angles are measured anticlockwise from this.

Often, the absolute magnitude is the only output the user sees --- the two components of the gradient are conveniently computed and added in a single pass over the input image using the pseudo-convolution operator shown in Figure 2.4.

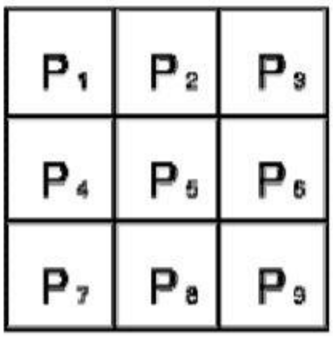


fig 2.4: pseudo convolution Sobel mask for fast computation

Using this mask the approximate magnitude is given by:

$$|G| = |(P_1 + 2 * P_2 + P_3) - (P_7 + 2 * P_8 + P_9)| \\ + |(P_3 + 2 * P_6 + P_9) - (P_1 + 2 * P_4 + P_7)|$$

2.3. Prewitt operator

2.3.1. Brief description

The Prewitt operator is also another useful tool in image processing, especially in edge detection algorithms. Technically, it computes an approximation of the gradient of the image intensity function, we can call it as a discrete differentiation operator. At each point in the image, the result of the Prewitt operator is either the corresponding gradient vector or the norm of this vector. The Prewitt operator is based on convolving the image with a tiny, separable and integer valued filter in both the perpendicular orientations. Therefore, this operator is inexpensive in comparison with other operators in terms of computations like Sobel operators. One more noticeable feature of this operator is the gradient approximation by this operator is relatively crude, especially for high frequency variations in the image. The Prewitt operator was developed by Judith M. S. Prewitt.

Basically at each the gradient of the image intensity can be computed by this operator, where the direction of the largest possible increase from light to dark and the rate of change in that direction. Therefore the outcome highlights how rapidly or continuously the change occurs at each point. Hence, we can understand how much that part of the image is likely to represent an edge, also how that particular edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation.

Mathematically, the gradient of a two-variable function (here the image intensity function) is at each image point a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each point of the image, the gradient vector shows us the direction of largest possible intensity increase, whereas the length of the gradient vector is related to the rate of change in that direction. This implies that the result of the Prewitt operator at an image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values.

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Fig 2.5: Prewitt operator

2.3.2. How it works

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. It is similar to Sobel operator. The *x*-coordinate is defined here as increasing in the "left"-direction, and the *y*-coordinate is defined as increasing in the "up"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$|G| = ({G_x}^2 + {G_y}^2)^{\frac{1}{2}}$$

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

In this case, orientation 0 is taken to mean that the direction of maximum contrast from black to white runs from left to right on the image, and other angles are measured anticlockwise from this.

The Prewitt masks are simpler than the Sobel masks, but, there is a slight computational difference between them which is not an issue The important fact is the Sobel masks have better noise suppression (smoothing) characteristics, which makes them preferable. Since, noise suppression is an important when we are working with derivatives [8].
Remarkable point is, unlike the Sobel operator, this operator does not place any emphasis on pixels that are closer to the center of the masks.

$$G_x = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Prewitt mask for detecting diagonal edges

However this implementation is not always suitable since the computational part here is laborious. A common approach we use here is approximation of the magnitude of the gradient by absolute values:

$$|G| = |G_x| + |G_y|$$

2.4. The canny edge detector

2.4.1. Brief description

The Canny operator was prepared to be an optimal edge detector (according to particular criteria --- there are other detectors around that also claim to be optimal with respect to slightly different criteria). It takes as input a grey scale image, and produces as output an image showing the positions of tracked intensity discontinuities.

Canny edge detection is a suitable for extracting necessary structural data from different images and minimizes the quantity of data required to be processed in large extent. It has been used widely in various computer vision systems. Canny has found that the requirements for the application of edge detection on diverse vision systems are relatively similar. Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include:

1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

To satisfy these requirements Canny used the calculus of variations – a technique which finds the function which optimizes a given functional. The optimal function in Canny's detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian.

$$\frac{d}{dx}e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}}$$

Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the three criteria for edge detection and the simplicity of process for implementation, it became one of the most popular algorithms for edge detection.

2.4.2. Process for Canny edge detection algorithm

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges

The Canny Edge Detector has its basis in a slightly more visual approach. If one considers a one-dimensional step edge change in contrast and then convolves that edge with a Gaussian smoothing function, the result will be a continuous change from the initial to final value, with the slope reaching a maximum at the point of the original step. If this continuous slope is then differentiated with respect to x, this slope maximum will become the maximum of the new function again at the point of the original step. Because the derivative of the convolution of the Gaussian and the image is the same as the convolution of the derivative of the Gaussian and the image. A mask can be created that represents the first derivative of the Gaussian. The maximums of the convolution of the mask and the image will indicate edges in the image. This process can be accomplished through the use of a two dimensional Gaussian function or the combination of a one-dimensional Gaussian function in both the x- and the y-directions. The values of the differentiated Gaussian mask depend on the choice of sigma in the Gaussian equation. The computational intensity of the Canny Edge Detector is relatively high, and the results are usually post-processed for clarity. However, the algorithm is very effective in processing noisy data or images with fuzzy edges [19] [20].

2.4.3. How it works

The Canny operator works in a multi-stage process. First of all the image is smoothened by Gaussian convolution. Then a simple 2-D first derivative operator (somewhat like the Roberts Cross) is applied to the smoothened image to focussed regions of the image with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as nonmaximal suppression. The tracking process exhibits hysteresis controlled by two thresholds: $T1$ and $T2$ with $T1 > T2$. Tracking can only begin at a point on a ridge higher than $T1$. Tracking then continues in both directions out from that point until the height of the ridge falls below $T2$. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.

2.5. Laplacian of Gaussian edge detector

2.5.1. Brief description

The zero crossing detector looks for places in the Laplacian of an image where the value of the Laplacian passes through zero --- i.e. points where the Laplacian changes sign. Such points often occur at `edges' in images --- i.e. points where the intensity of the image changes rapidly, but they also occur at places that are not as easy to associate with edges. It is best to think of the zero crossing detector as some sort of feature detector rather than as a specific edge detector. Zero crossings always lie on closed contours and so the output from the zero crossing detector is usually a binary image with single pixel thickness lines showing the positions of the zero crossing points. The starting point for the zero crossing detector is an image which has been filtered using the Laplacian of Gaussian filter. The zero crossings that result are strongly influenced by the size of the Gaussian used for the smoothing stage of this operator. As the smoothing is increased then fewer and fewer zero crossing contours will be found, and those that do remain will correspond to features of larger and larger scale in the image.

2.5.2. How it works

The basics of the zero crossing detector is the Laplacian of Gaussian filter and So a knowledge of that operator is assumed here. As described there, `edges' in images give rise to zero crossings in the LoG output.

However, zero crossings are also noticed at any place other than edges. We can observe zero crossings where the gradient of image intensity gradient increases decreases. Often zero crossings are observed in the regions of very low gradient i.e. where the intensity gradient moves up and down around zero.

Once the image has been LoG filtered, we are only left with the detection of the zero crossings. This can be performed in numerous ways.

The simplest process for performing the above is to simply threshold the LoG output at zero, producing a binary image where the boundaries between foreground and background regions represent the locations of zero crossing points. These boundaries can then be easily identified and marked in single pass, e.g. using some morphological operator. For instance, to detect all boundary points, we simply have to mark each foreground point that has at least one background neighbour. The problem with this technique is that will tend to bias the location of the zero crossing edge to either the light side of the edge, or the dark side of the edge, depending upon whether it is decided to look for the edges of foreground regions or for the edges of background regions. A better technique is to consider points on both sides of the threshold boundary, and choose the one with the lowest absolute magnitude of the Laplacian, which will hopefully be closest to the zero crossing. Since the zero crossings generally fall in between two pixels in the LoG filtered image, an alternative output representation is an image grid which is spatially shifted half a pixel across and half a pixel down relative to the original image. Such a representation is known as a dual lattice. This does not actually localize the zero crossing any more accurately of course. A more accurate approach is to perform some kind of interpolation to estimate the position of the zero crossing to sub-pixel precision.

Chapter 3

Stochastic Architecture of Edge detection

Stochastic Architecture of Edge detection

So far we have discussed about the theoretical concept of the edge detection algorithm. In this chapter we will discuss about the implementation of the edge detector using the stochastic computing. We will elaborate the stochastic architecture of the edge detection. We have discussed many algorithm in the last chapter like Robert's, Sobel, Prewitt, Canny, and Laplacian of Gaussian edge detector. Here, in this chapter we will see the stochastic architecture of the Robert's edge detector. All the stochastic element, used in this stochastic structure of the Robert's edge detector will be explained in the next chapter.

3.1. Stochastic Architecture for Robert's Edge Detection

Edge detection is the most frequently used method for segmenting images based on abrupt changes in intensity. In the image processing, edge detection is one of the essential weapon or tool which tells us the location of abrupt change or sudden change of the intensity level in the image and it creates an edge at that location of discontinuity of brightness. There are many number of edge detection operators available, each are introduced to be sensitive to certain types of edges. The conventional method of edge detection introduce convolution of an image with a filter, which is constructed to be sensitive to large gradients in the image while returning values of zero in the uniform regions[8]. Here we are using Robert's cross operators .Robert's cross operators are one of the earliest method using 2-D masks with a diagonal preference. The Roberts operators are based on implementing the diagonal differences.

Let $x_{i,j}$ denotes the pixel value at location (i,j) of the original image and $y_{i,j}$ is the pixel intensity value at location (i,j). The convolution between the original image and the Robert's cross operator and produce output as given as below

$$y_{i,j} = \frac{1}{2}\left[\left|x_{i+1,j+1} - x_{i,j}\right| + \left|x_{i,j+1} - x_{i+1,j}\right|\right] \qquad (1)$$

From the above equation we can say we need stochastic element such as scaled addition, subtraction absolute value.

+1	0
0	-1

G_X

0	+1
-1	0

G_Y

fig 3.1: Robert's cross operator

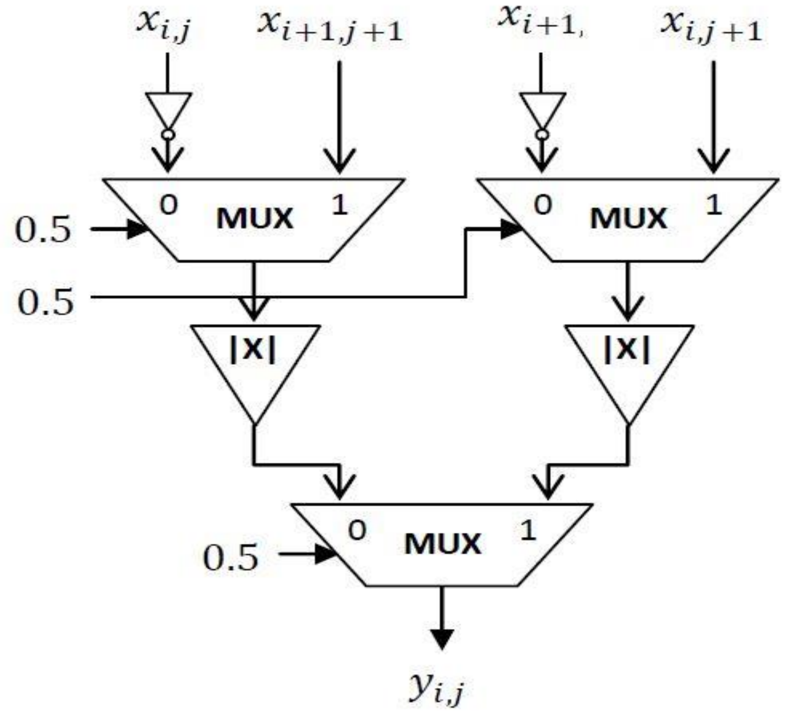


fig 3.2: Stochastic Architecture of Robert's Edge detector

In this architecture we have used stochastic scaled addition, absolute value, complement of probability value. Scaled addition operation is performed by a MUX, complement of probability is performed by an inverter and absolute value is done stochastic component based on FSM.

This architecture is the replica of the equation of the (1). Here original image is convoluted with the mask and result will store in the pixel as intensity with respect to the predefined threshold value and edge will be produced.

For the detail knowledge of this architecture we should know about the stochastic logic circuit about FSM.

3.2. Stochastic FSM

In this section, we present and analyse FSM-based stochastic computation elements (SCEs) for the STanh function, the SExp function, the stochastic ExpAbs (SExpAbs) function, and the stochastic Abs (SAbs) function. The constructs for the STanh and SExp functions were presented by Brown and Card [3]; they provided no proof of correctness, only empirical validation. The constructs for the SExpAbs and SAbs functions are new. We prove the correctness of all four constructs. Combinational logic can only implement polynomial functions of a specific form – namely those that map the unit interval to the unit interval [21]. Non-polynomial functions can be approximated by combinational logic, for instance with Maclaurin expansions [22]. However, highly nonlinear functions such as exponentiation and tanh cannot be approximated effectively because of the fact that combinational logic can only implement scaled addition. The function which require simple arithmetic operations can be implemented stochastically by using combinational gates. But there are some other functions which requires complex hardware and cannot be achieved by using sequential circuits, such as computing absolute value and to derive hyperbolic functions, and so cannot be easily implemented with these combinational logic gates. Fortunately, stochastic computational elements based on finite state machine (FSM) can resolve this issue. The main objective of this work is to develop stochastic computational elements to generate stochastic functions of the particular inputs and it depends on the control inputs. In these stochastic constructs the state transition will take place depends on the characteristics of the function and the set of the state variables. Brown and Card [3] and Li et al. [23] have presented linear FSM based computing elements for stochastic implementation of such complex functions. They used linear state machine based on Markov model to implement hyperbolic-tangent, exponentiation, absolute-value, exponentiation with absolute-value and linear gain functions. But, the linear FSM based approach cannot be applied to any given function in general. Later, Li et al. [24] proposed a new FSM (2-D) based approach to synthesize any target function with desired accuracy by framing it as a constrained quadratic programming problem. We discuss these implementations one by one. From the markov model properties, the sum of the probabilities of all the state is unity and the probability that a state transition occurs from the state S_0 to S_1. Mathematically all these conclusions are shown in the following discussions.

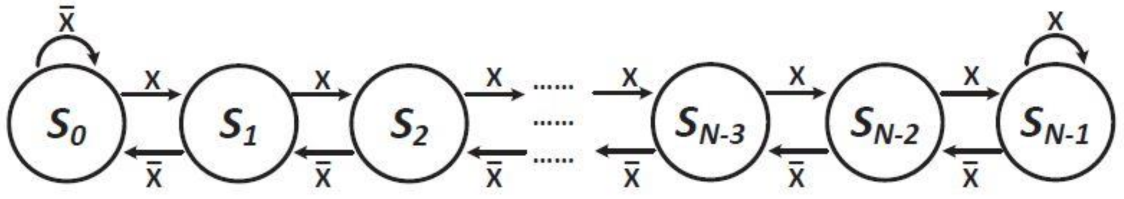


fig 3.3:linear stochastic state machine

It has $N = 2^M$ states, where M is a positive integer. X is the input bit stream of this state machine. Here the bit stream length and the number of states used in the design are independent. In this state machine the output bit stream Y is determined only by the states. Consider P_X is the probability of the output bit stream. At any particular instant if the present state S_i $(0 \le i \le N-1)$ for a particular input P_X then the probability of that state is denoted as $P_{i(P_X)}$. Then the output probability is given by

$$P_y = S_i P_{i(P_X)}$$

Where, the individual state S_i only has two values, either 0 or 1. For example, setting $S_1 = 1$ denotes $Y = 1$ if the current state is S_1. The system is random and will have one single stable state. The individual state transitions from a particular state must sum to unity, and the probability of transitioning from state S_{i-1}. Based on the above analysis we have

$$P_{i(P_x)}.(1 - P_x) = (P_{(i-1)(P_x)})P_x$$

$$\sum_{i=0}^{N-1} P_{i(P_x)} = 1$$

Furthermore, based on different intervals of P_x, $P_{i(P_x)}$can also be computed as follows

$$P_{i(P_x)} = \begin{cases} \left(\frac{P_x}{1-P_x}\right)^i . \left(\frac{1-2P_x}{1-P_x}\right) & P_x < .5 \\ \frac{1}{N} & P_x = .5 \\ \left(\frac{1-P_x}{P_x}\right)^{N-1-i} . \left(\frac{2P_x-1}{P_x}\right) & P_x > .5 \end{cases}$$

If we define

$$t_{(P_x)} = \frac{.5 - |P_x - .5|}{.5 + |P_x - .5|}$$

Then the above equation can be written as

$$P_{i(P_x)} = \begin{cases} \frac{t_{(P_x)}^i . (1 - t_{(P_x)})}{1 - t_{(P_x)}^N}, & 0 \le P_x \le .5 \\ \frac{1}{N}, & P_x = .5 \\ \frac{t_{(P_x)}^{N-1-i} . (1 - t_{(P_x)})}{1 - t_{(P_x)}^N} & .5 \le P_x \le 1 \end{cases}$$

Based on the above properties, we analyse and present some FSM-based stochastic computation elements.

3.2.1 Stochastic approximation method of Tanh function (STanh)

To implement nonlinear mapping STanh is used and based on the markov chain properties this function can be approximated stochastically. Here we are not considering the control variables.

This state S_i is as follows

$$S_i = \begin{cases} 0, & 0 \le i \le \frac{N}{2} - 1 \\ 1, & \frac{N}{2} \le i \le N - 1 \end{cases}$$

Proof:

Based on the configuration and properties the output probability is given by

$$P_Y = \sum_{i=\frac{N}{2}}^{N-1} P_{i(P_X)}$$

If we substitute $P_{i(P_X)}$ then we have,

$$P_Y = \frac{\sum_{i=\frac{N}{2}}^{N-1}\left(\frac{P_X}{1-P_X}\right)^i}{\sum_{k=0}^{N-1}\left(\frac{P_X}{1-P_X}\right)^k} = \frac{\left(\frac{P_X}{1-P_X}\right)^{\frac{N}{2}} - \left(\frac{P_X}{1-P_X}\right)^N}{1-\left(\frac{P_X}{1-P_X}\right)^N}$$

$$= \frac{\left(\frac{P_X}{1-P_X}\right)^{\frac{N}{2}}.(1-\left(\frac{P_X}{1-P_X}\right)^{\frac{N}{2}})}{\left(1+\left(\frac{P_X}{1-P_X}\right)^{\frac{N}{2}}\right).(1-\left(\frac{P_X}{1-P_X}\right)^{\frac{N}{2}})} = \frac{\left(\frac{P_X}{1-P_X}\right)^{\frac{N}{2}}}{\left(1+\left(\frac{P_X}{1-P_X}\right)^{\frac{N}{2}}\right)}$$

Here input and output are defined in bipolar coding format, therefore corresponding probabilities are substituted as

$$\frac{y+1}{2} = \frac{\left(\frac{1+x}{1-x}\right)^{\frac{N}{2}}}{1+\left(\frac{1+x}{1-x}\right)^{\frac{N}{2}}}$$

Simplifying

$$y = \frac{\left(\frac{1+x}{1-x}\right)^{\frac{N}{2}} - 1}{\left(\frac{1+x}{1-x}\right)^{\frac{N}{2}} + 1}$$

By using Taylor's expansion,

$$1+x \approx e^x$$
$$1-x \approx e^{-x}$$

Therefore

$$y = \frac{(e^{2x})^{\frac{N}{2}} - 1}{(e^{2x})^{\frac{N}{2}} + 1} = \frac{e^{\frac{N}{2}x} - e^{-\frac{N}{2}x}}{e^{\frac{N}{2}x} + e^{-\frac{N}{2}x}}$$

3.2.2 Stochastic exponentiation function (SExp):

In the SExp the state values are represented as follows,

$$S_i = \begin{cases} 1, & 0 \le i \le N - G - 1 \\ 0, & N - G \le i \le N - 1 \end{cases}$$

Where G is a positive integer and $G \ll N$. We have shown this configuration in fig 3.4 and equation is approximate to the exponentiation function. We will prove as follows.

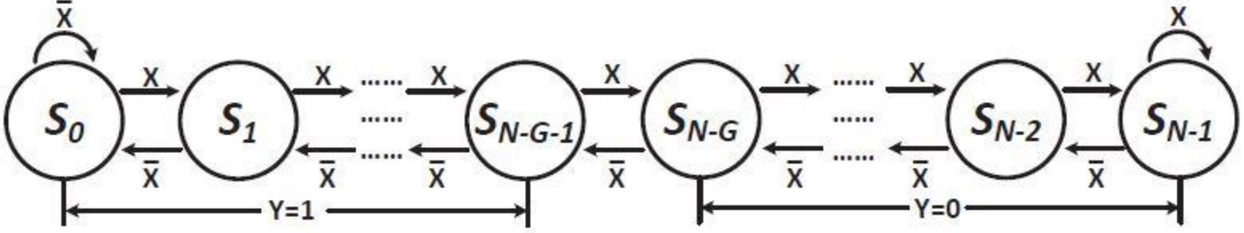


fig 3.4 : State transition diagram of the FSM based stochastic exponentiation function

Proof :

Based on the above configuration and properties, the output probability is given by

$$P_Y = \sum_{i=0}^{N-G-1} P_{i(P_x)}$$

If we substitute $P_{i(P_x)}$, we have

$$P_Y = \begin{cases} \displaystyle\sum_{i=0}^{N-G-1} \frac{t_{(P_x)}^{i} . (1 - t_{(P_x)})}{1 - t_{(P_x)}^{N}}, & 0 \le P_x \le .5 \\ \dfrac{N - G}{N}, & P_x = .5 \\ \displaystyle\sum_{i=0}^{N-G-1} \frac{t_{(P_x)}^{N-1-i} . (1 - t_{(P_x)})}{1 - t_{(P_x)}^{N}}, & .5 \le P_x \le 1 \end{cases}$$

Based $G \ll N$ we can write as follows,

$$P_Y \approx \begin{cases} \displaystyle\sum_{i=0}^{\frac{N}{2}-1} \frac{t_{(P_x)}^{i}.(1-t_{(P_x)})}{1-t_{(P_x)}^{N}}, & 0 \le P_x \le .5 \\ 1, & P_x = .5 \\ \displaystyle\sum_{i=0}^{N-G-1} \frac{t_{(P_x)}^{N-1-i}.(1-t_{(P_x)})}{1-t_{(P_x)}^{N}}, & .5 \le P_x \le 1 \end{cases}$$

1. When$P_X > .5, t_{(P_x)} = \frac{P_x}{1-P_x} < 1$. When N is large enough, we have,

$$\sum_{i=0}^{\frac{N}{2}-1} \frac{t_{(P_x)}^{i}.(1-t_{(P_x)})}{1-t_{(P_x)}^{N}} = \frac{1-t_{(P_x)}^{\frac{N}{2}}}{1-t_{(P_x)}^{N}} \approx 1$$

2. When$P_X > .5, t_{(P_x)} = \frac{1-P_x}{P_x}$. When N is large enough, we have,

$$\sum_{i=0}^{N-G-1} \frac{t_{(P_x)}^{N-1-i}.(1-t_{(P_x)})}{1-t_{(P_x)}^{N}} = \frac{t_{P_x}^{G} - t_{P_x}^{\frac{N}{2}}}{1-t_{(P_x)}^{N}} \approx t_{P_x}^{G}$$

Because $x = 2P_x - 1$ and $P_X > .5$, we can rewrite in as follows,

$$t_{(P_x)} = \frac{1-|x|}{1-|x|} = \frac{1-x}{1+x}$$

By using Taylor's expansion, we have

$$1 + x \approx e^{x}$$
$$1 - x \approx e^{-x}$$

So,

$$t_{(P_x)} = \frac{1-|x|}{1-|x|} = \frac{1-x}{1+x} \approx \frac{e^{-x}}{e^{x}} = e^{-2x}$$

Therefore

$$t_{P_x}^{G} = e^{-2x}$$

3.2.3. SExp with an Absolute Value (SExpAbs)

The set of states can be represented as

$$S_i = \begin{cases} 1, & G \le i \le N - G - 1 \\ 0, & i < G \text{ or } i > N - G - 1 \end{cases}$$

We also show this configuration in fig 3.5. The approximate transfer function in this configuration is

$$P_Y = e^{-2G|x|}$$

Where x is bipolar coding of P_x , i.e., $x = 2P_x - 1$.

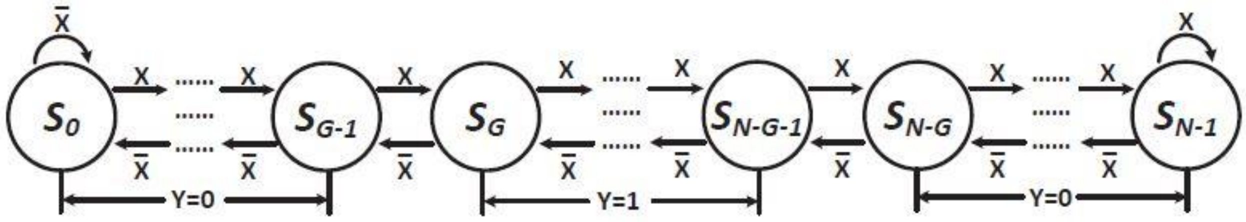


Fig 3.5: State transition diagram of the FSM based stochastic exponential absolute function

Proof:

1. Where input probability $P_x > .5(i.e., x > 0)$, the configuration from $S_{N/_2}$ to S_{N-1} is the same. Based on properties, the configuration has the same function as in when $P_x > .5$. thus we have

$$P_Y = e^{-2Gx}$$

2. When $P_x < .5(i.e., x < 0)$, because $S_i = S_{N-1-i}$, based on property, P_Y is symmetric about $P_x = .5$, i.e. P_Y is symmetric about $x = 0$. Thus we have

$$P_Y = e^{2Gx}$$

As a result, on, we obtain, note that, since $G \ll N$, when $P_x = .5$, $(i.e. x = 0)$, based on,

$$P_Y = \frac{N - 2G - 1}{N} \approx 1$$

3.4 . Stochastic Digital Implementation of Non-Linear Functions

The markov chain properties and the FSM diagrams that we discussed previous are used in this design. This is a common design for all the functions where there are some differences in representing the state values (either 0 or 1) as in FSM implements all the four functions based on the state values. The SNG is used to convert to bit stream. At every clock pulse depends on the bit it will count either up or down the N-bit U/D counter. If the bit is 1 it will count UP or if the bit is) it will count DOWN. For all the functions we used N=32 state FSM. So after every clock pulse the counter will stay in a particular state. As shown in the state diagrams each state is represented with either 1 or 0. Depends on the state number the bit will go as an input to the binary counter. Here we have to consider two conditions depend on the state number. At every clock pulse these conditions will check whether it is first or $(N-1)th$ state. For example if we are using 32 states then when present state is $0th$ state and the input bit stream is 0 then it should stay in that state only and another condition is for if the present state is last one and input is 1 then also it has to stay in that state that means there should not occur any state transitions. In this way the iterations will continue depends on bit stream length (in all the cases we considered 1024 bits). At every clock pulse the binary counter will increment depends on the bit and after all iterations it will give us a binary value where the output probability is the ratio of the binary count to the bit stream length.

The state values for the above functions are shown below:

Here we consider $N = 32, G = 2$, bit stream length $L = 1024$

STanh : [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

SExp : [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0]

SExpAbs : [0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0]

Sabs : [1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1]

Chapter 4

Realization of Stochastic Edge Detector

Realization of Stochastic Edge Detector

So far we have seen the stochastic architecture of the edge detector using Robert's operator. In this section we will discuss about the stochastic component used in the architecture and other basic stochastic computing element. At first we will start from the basic stochastic element and then we will move to the component used in the component used in the Robert's edge detector.

In this chapter we will discuss about the LFSR as a pseudorandom number generator. In this chapter we use this pseudorandom number generator to generate stochastic numbers (or bit-streams). These stochastic numbers need to be converted back into ordinary binary numbers at the end of stochastic processing. Thus, we need a de-randomizer unit (DRU).

4.1 Data representation

In the stochastic computer arithmetic operations are performed by virtue of the completely random and uncorrelated nature of the logic levels representing data, and only probability that a logic level will be ON or OFF is determined, its actual value cannot predicted and it is a chance event, and repetition of a computation will give rise to a dissimilar sequence of logic levels. Thus a quantity in the stochastic computer is represented by a clocked sequence of logic levels generated by a random process such that consecutive levels are statistically independent, and the probability of the logic level being ON is a measure of that quantity, such a sequence is called a Bernoulli sequence. The computation of the stochastic computer is achieved through its uncharacteristic representation of data by the probability that a logic level will be HIGH (logic 1) or LOW (logic 0) at a clock pulse. In any other form of computer the logic levels representing data change deterministically from value to value as the computation proceeds, and if the computation is repeated the same sequence of logic levels occur.

The physical certainty behind the mathematical concept of probability is, of course, that the generating probability of a sequence of logic levels corresponds to the relative frequency of ON logic levels in a sufficiently long sequence. In its use of relative frequency to convey information the stochastic computer is similar to the other incremental, or counting computers, operational digital computer, and a phase computer. In all these computers, including the stochastic computer, quantities are represented as binary words for purpose of storage, and as the proportion of ON logic levels in a clocked sequence for purposes of computation. In conventional machines, however, the sequences of logic levels are generated deterministically and are generally patterned, whereas in the stochastic computer each logic level is generated by a statistically independent process.

The stochastic data does not depend on the stream length or the position of the 1’s in the bit stream. Different bit stream can represent the same data, length of the bit stream or the position of the $1's$ need not to be fixed. For example (0100), $(0010), (1000), (0001)$ are all possible representation of $.25$. The probability depends on the ratio of number of $1's$ and bit stream length L, not on their positions so that we can represent randomly.

4.2 Coding format

In the stochastic encoding, different coding formats are present to represent the stochastic number by bit stream, like unipolar coding format, bipolar coding format and two line bipolar, single line bipolar. These two coding formats are same in essence, and coexist in a single system.

4.2.1 Unipolar

In the unipolar coding format, a real number x in the unit interval $(i.e. 0 \leq x \leq 1)$ corresponds to a bit stream X (t) of length L, where $t = 1,2, \dots, L$.
The probability that each bit in the stream is one is

$$P(X = 1) = x$$

For example, the value $x = .5$ would be represented by a random stream of bits such as 00110101, where approximately 50% of the bits are $1's$ and rest are $0's$.

4.2.2 Bipolar

In the bipolar coding format, the range of a real number x is extended to $-1 \leq x \leq 1$. However, the probability that each bit in the stream is one, is

$$P(X = 1) = \frac{(X + 1)}{2}$$

And we can define real number as

$$x = 2 * P(X = 1)$$

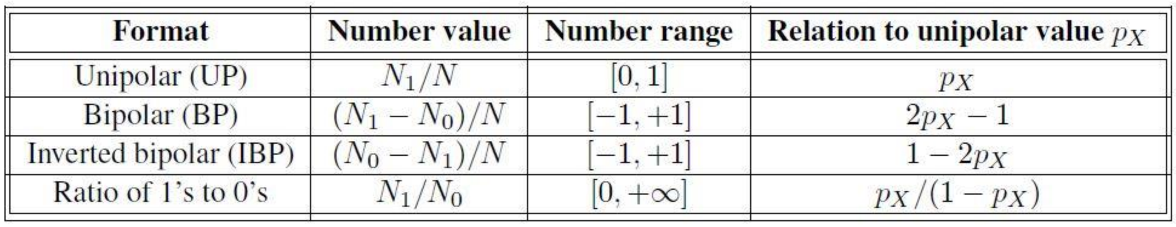
The difference between these two coding format is that the bipolar format will help to work with negative numbers directly while, given the same bit stream length, L, the exactness of the unipolar format is twice that of the bipolar format. For example to represent a number $-.5$, then probability will be $P = (-.5 + 1)/2 = .25$. Therefore 25% bits of the bit stream should be $1's$ and rest are $0's$. Similarly to represent $.5$ then in bipolar format, the probability will be $P = .75$, therefore 75% bits of the bit streams are $1's$ and rest are $0's$.

4.2.3 Two line

In this scheme, two bit-streams, instead of one, are used to interpret a number x. More specifically, the magnitude and sign information of x, are carried by a magnitude stream and a sign stream, respectively [15, 16]. If we denote $M(Xi)$ and $S(Xi)$as the ith bit of the magnitude stream and sign stream with length L, respectively, then x that is within $[-1,1]$ can be represented as

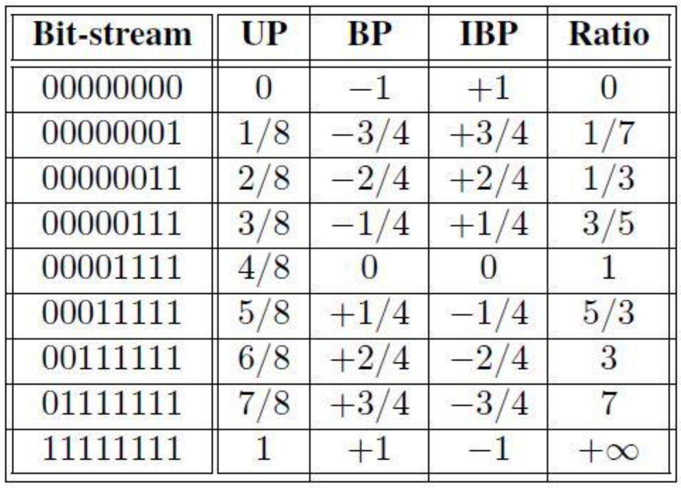
$$x = (1/L) \sum_{i=0}^{L-1} (1 - S(Xi))M(Xi)$$

Although the two-line representation introduces additional hardware for arithmetic units, this representation has a unique advantage for designing scaling-free stochastic adder.



Format	Number value	Number range	Relation to unipolar value p_X
Unipolar (UP)	N_1/N	$[0, 1]$	p_X
Bipolar (BP)	$(N_1 - N_0)/N$	$[-1, +1]$	$2p_X - 1$
Inverted bipolar (IBP)	$(N_0 - N_1)/N$	$[-1, +1]$	$1 - 2p_X$
Ratio of 1's to 0's	N_1/N_0	$[0, +\infty]$	$p_X/(1 - p_X)$

Table 4.1: Stochastic value for different format



Bit-stream	UP	BP	IBP	Ratio
00000000	0	-1	$+1$	0
00000001	$1/8$	$-3/4$	$+3/4$	$1/7$
00000011	$2/8$	$-2/4$	$+2/4$	$1/3$
00000111	$3/8$	$-1/4$	$+1/4$	$3/5$
00001111	$4/8$	0	0	1
00011111	$5/8$	$+1/4$	$-1/4$	$5/3$
00111111	$6/8$	$+2/4$	$-2/4$	3
01111111	$7/8$	$+3/4$	$-3/4$	7
11111111	1	$+1$	-1	$+\infty$

table 4.2: Representation of bit stream in different stochastic format

4.3 Generation of stochastic Numbers

The input data to all the processing elements are the stochastic bit streams. This means that the probability of a given bit being 1 is independent of the values of any previous bits. The bit representation is random here. For any input value we can represent in bit stream of different order independent of any position. There are few method where we can convert the input data to the bit stream. These method includes linear feedback shift register (LFSR).

4.3.1 Linear Feedback Shift Register (LFSR)

The linear feedback shift register (LFSR) is a shift register which, using feedback, modifies itself on each rising edge of the clock. The feedback causes the value in the shift register to cycle through a set of unique values. The choice of LFSR length, gate type, LFSR type, maximum length logic, and tap positions allows the user to control the implementation and feedback of the LFSR, which, in turn, controls the sequence of repeating values the LFSR will iterate through. The LFSR can be created using the Fibonacci configuration of gates and registers. In the Fibonacci implementation, the outputs from some of the registers are exclusive-ORed with each other and feedback to the input of the shift register. Fig 4.1 shows a 4-bit Fibonacci LFSR. When the shift register is loaded with a seed value (any value except all zeros) and then clocked, the output from the LFSR will be a pseudo-random sequence of $1's$ and $0's$. For example, table 4.4 shows the sequence that is produced from the circuit in fig 4.1 when the seed is 1010. The length of the pseudo-random sequence is dependent on the length of the shift register and the number and the position of the feedback taps.

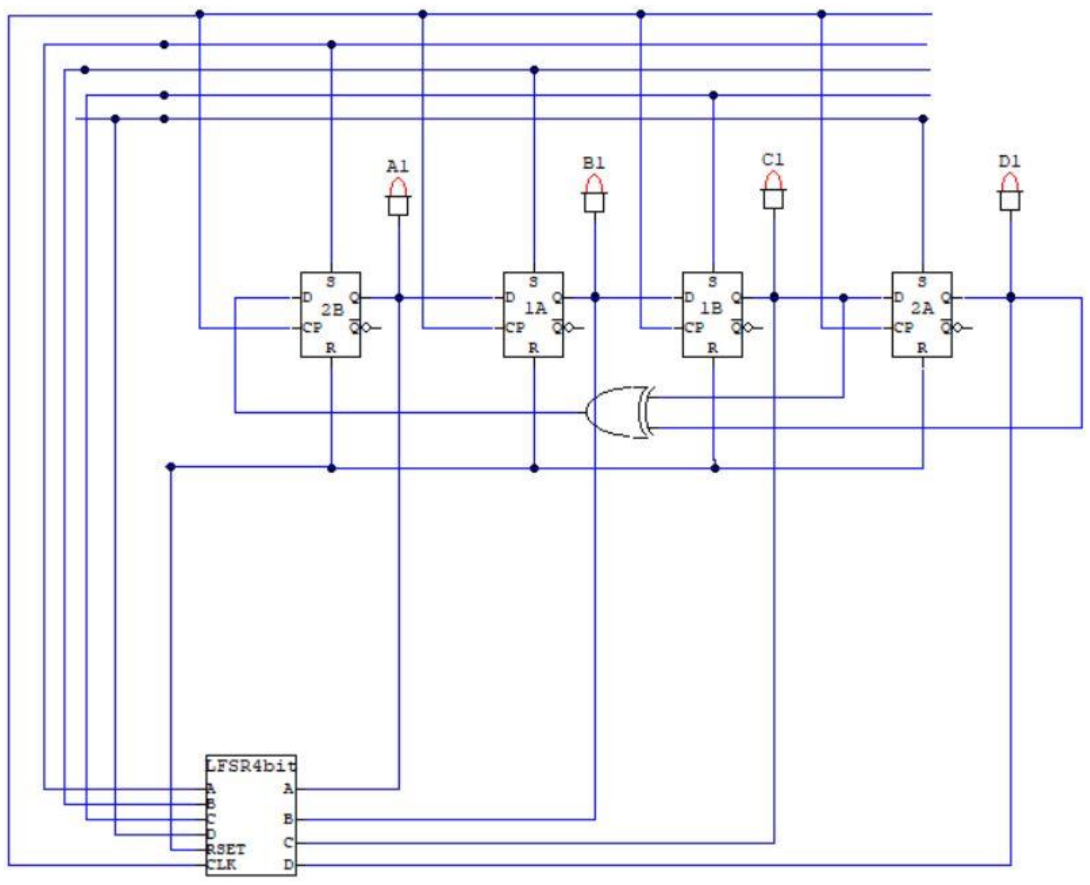


fig 4.1: 4bit LFSR

In the figure 4.1 shows the design of 4-bit LFSR using the XOR gate for feedback path. For the different position of the tap position will create the different length of cycle of pseudorandom sequences to be generated by LFSR. We should select the tap position for which we can get the maximum length of period. For an n-bit LFSR, the maximum length or period is $2^n - 1$. Table 4.3 shows the feedback positions or taps for maximal length cycle of pseudorandom sequences to be generated by n-bit LFSR.

Bits(n)	Characteristic polynomial	Period($2^n - 1$)
2	D2+D+1	3
3	D3+D+1	7

4	D_4+D_3+1	15
5	D_5+D_3+1	31
7	D_7+D_6+1	127
8	$D_8+D_6+D_5+D_4+1$	255
10	$D_{10}+D_7+1$	1023
19	$D_{19}+D_{18}+D_{17}+D_{14}+1$	524287

Table 4.3: maximal-length Feedback polynomials for n-bit LFSR

Q_3	Q_2	Q_1	Q_0
1	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
0	1	0	1
1	0	1	0

Table 4.4: sequence of the LFSR in fig 4.1 using seed 1010

4.3.2. Stochastic interface

SNG and DRU are together works as Stochastic Interface between the binary input and the stochastic representation of binary values. SNG generates stochastic bit stream from the input whereas DRU returns the output as binary value from the stochastic value.

4.3.2.1. Stochastic Number generator (SNG)

SNG is consists of LFSR, a register and a binary comparator. The reason of using SNG is to generate the stochastic bit streams from a binary value. In this process a binary value X is stored in the register, and LFSR is continuously generating a binary random streams for every clock pulse. In every clock pulse these two streams are compared in the binary comparator and produce a single Y_i bit either 1 or 0. After $L = (2^n - 1)$ clock pulses we get the stochastic representation of A in the form of an L-bit Bernoulli sequence Y.

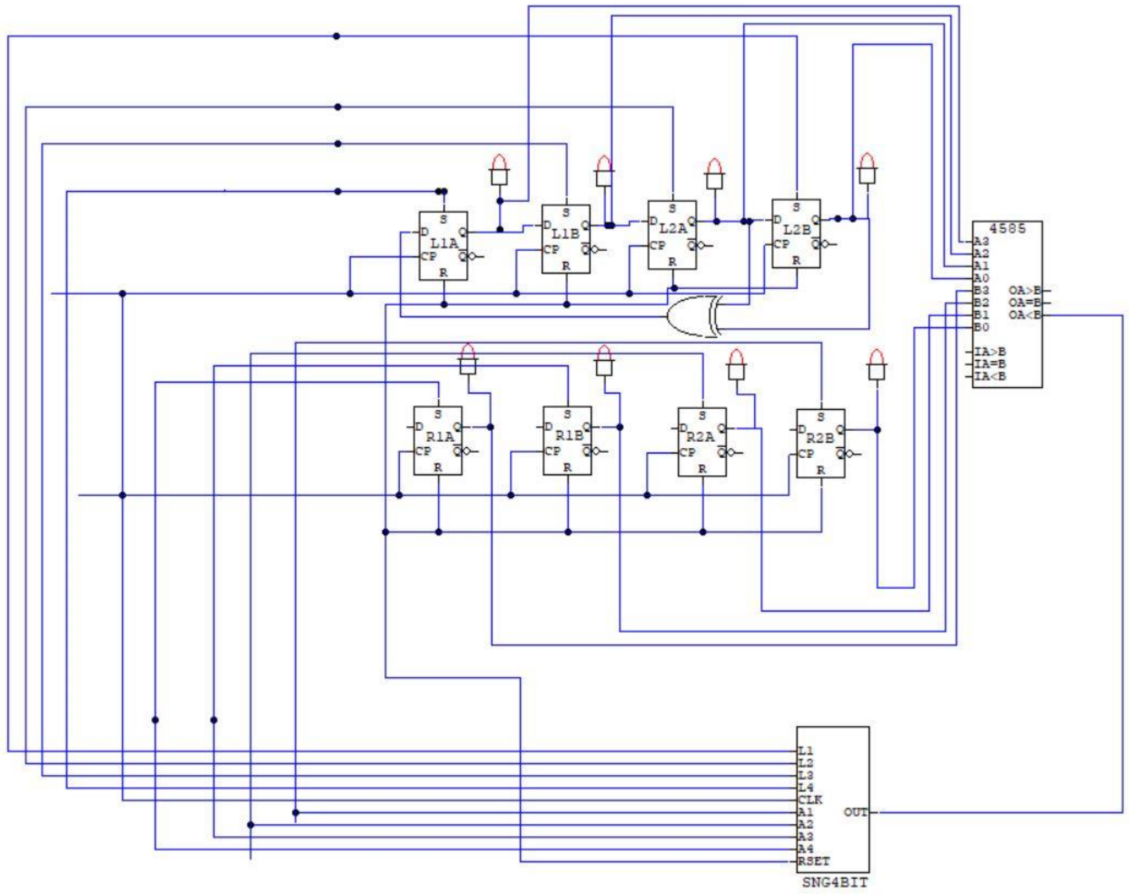


fig 4.2: 4-bit SNG

In the SNG, comparator compares LFSR output bit stream and Register value stream in a single clock pulse.

4.3.2.1 De-Randomizer Unit (DRU)

Results should be always in a manner or in a format where we can easily track it or accept it. So at the end of the stochastic computing we have to convert the stochastic value in our familiar representation that is in conventional binary representation. DRU is used for this purpose. That's why it is part of the stochastic interface. It consists of a binary up counter which counts the no of $1's$ in the output bit stream Y and after $L = (2^n - 1)$ clock pulses we will get the required binary value.

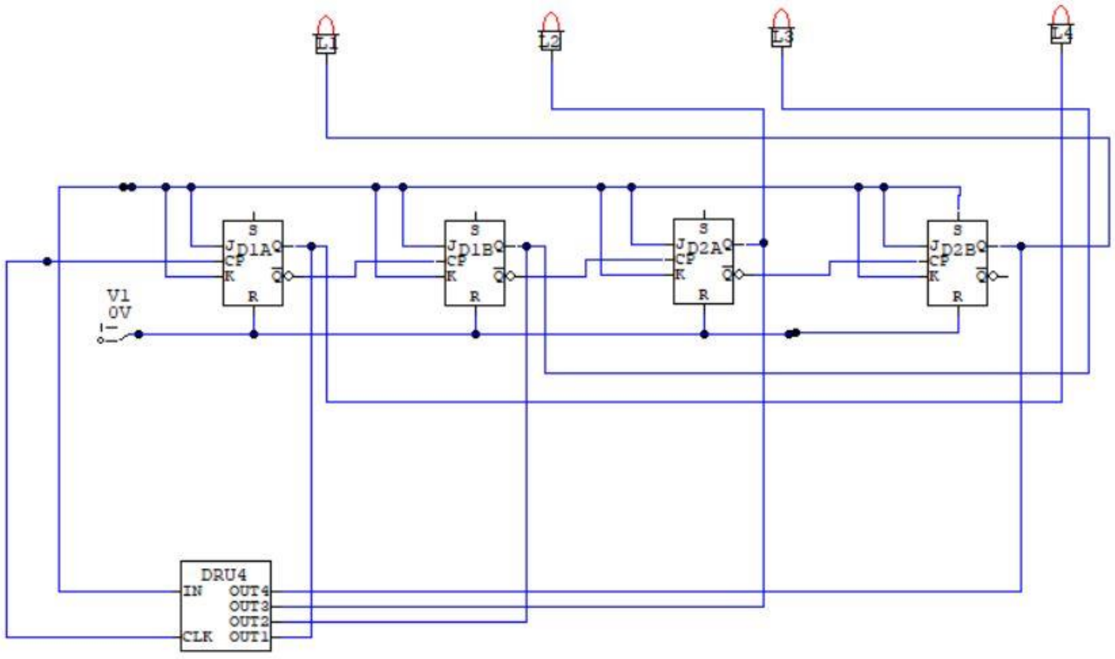


fig 4.3: 4-bit DRU

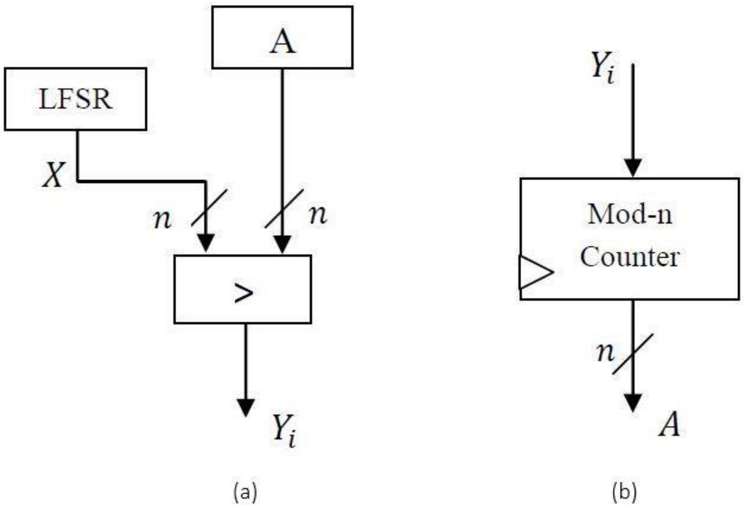


fig 4.4: Schematic diagram of (a) SNG (b) DRU

4.4. Circuits realization of stochastic component used in Robert’s edge detection

Here we are going to discuss about those components which are used in the implementation of the Robert’s edge detector. All this components are implemented by stochastic logic.

4.4.1. Invertor

Using a single NOT gate we can get complement of any probability value. Suppose A is an event with occurrence of probability X. Then the not occurrence of A is denoted by the probability value of $(1-X)$. So if X represents the stochastic value x then complement of X will be represented by $(1-x)$.

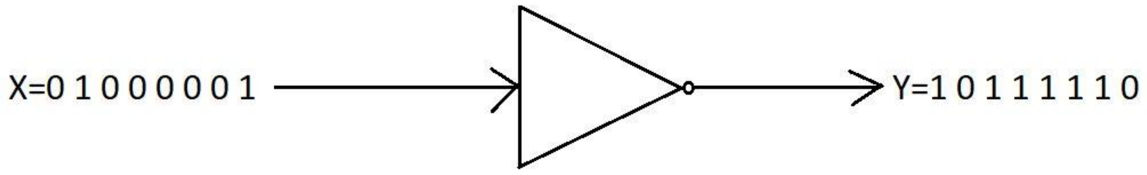


fig 4.5: Stochastic invertor using NOT gate

For example, for $x = \frac{2}{8}$, $X = 01000001$. Now $X' = 10111110$ will represent $(1-x) = \frac{5}{8}$ as shown in the figure 4.5.

4.4.2. Scaled addition

Using Multiplexer and stochastic no generator we can implement stochastic scaled addition. This is a scaled version of an addition performance of two number. This is scaled to get the result in the range of [0,1]. In many polynomial there are a coefficient which is less than one, in that case scaled addition may be used. This scaling factor must be input of the select line of the MUX as shown in the fig 4.6. The principle of operation here is the selection input selects one bit at every clock pulse from any one of the given input.

Scaled addition is designed by using a multiplexer, where the inputs and the select line of the multiplexer are stochastic bit stream generated by using SNG.

Assume A,B,C and S are stochastic bit streams, and $P_A, P_B, P_C \; and \; P_S$ are their related probabilities obtained by taking the ratio of ones to the length of the bit stream L, then we get

$$P_C = P_A * P_S + (1 - P_S) * P_B$$

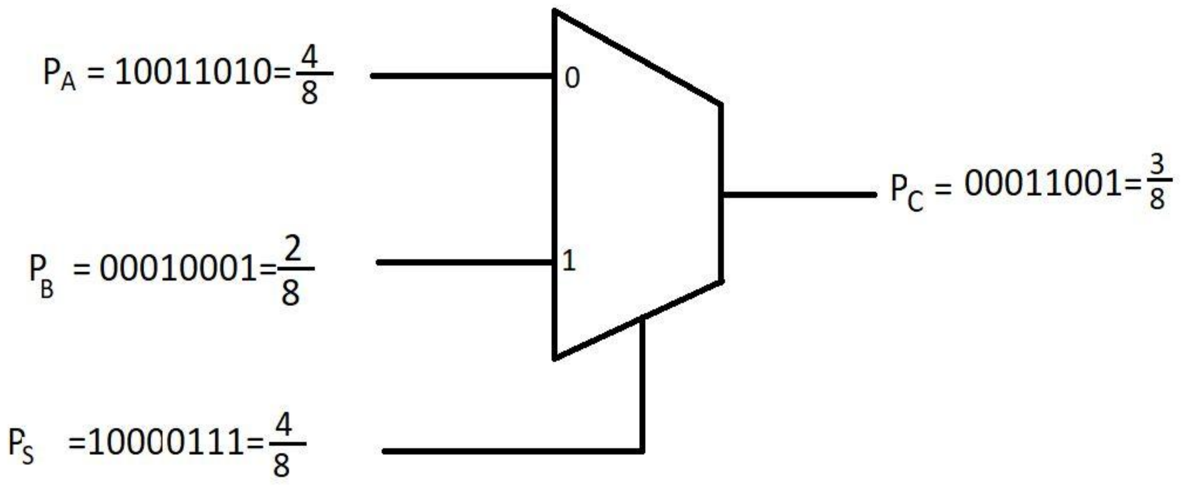


fig 4.6: stochastic scaled addition

4.4.3. Stochastic Absolute value function based on FSM (SAbs)

Here the set of states are approximated as,

When $0 \leq i \leq \frac{N}{2} - 1$,

$$S_i = \begin{cases} 1, & i \text{ is even} \\ 0, & i \text{ is odd} \end{cases}$$

When $\frac{N}{2} \leq i \leq N$,

$$S_i = \begin{cases} 0, & i \text{ is even} \\ 1, & i \text{ is odd} \end{cases}$$

Fig 3.16 show, the configuration based on a N-state FSM. If we define x as the bipolar coding of P_x and y as the bipolar coding of P_Y. The approximate transfer function in this configuration is

$$y = |x|.$$

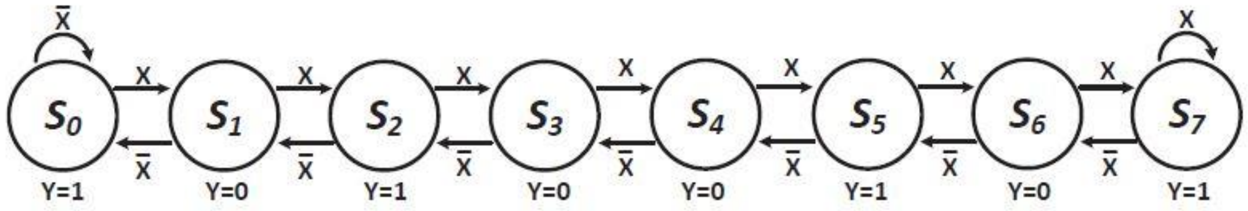


Fig 3.7 : State transition diagram of the FSM based absolute function

Proof :

Based on the configuration in, we have

$$P_Y = \sum_{i=0}^{\frac{N}{4}-1} P_{2i(P_x)} + \sum_{i=0}^{\frac{N}{4}-1} P_{(2i+1)(P_x)}$$

1. When $P_x < .5$, based on property, we have

   $$P_Y \approx \sum_{i=0}^{\frac{N}{4}-1} P_{2i(P_x)}$$

   Substituting $P_{2i(P_x)}$ and $t_{(P_x)}$ successively we get

   $$P_Y \approx 1 - P_x.$$

   So

   $$y = 2P_Y - 1 = 2(1 - P_x) - 1 = -x$$

2. When $P_x < .5(i.e., x < 0)$, because $S_i = S_{N-1-i}$, based on property, P_Y is symmetric about $P_x = .5$, i.e.

   $$P_Y \approx 1 - (1 - P_x) = P_x.$$

   Thus

   $$y = 2P_Y - 1 = 2(P_x) - 1 = x.$$

3. When $P_x = .5$, based on $P_{i(P_x)} = \frac{1}{N}$. if we substitute $P_{i(P_x)}$ with $\frac{1}{N}$, we have

   $$P_Y = \sum_{i=0}^{\frac{N}{4}-1} \frac{1}{N} + \sum_{i=\frac{N}{4}}^{\frac{N}{2}-1} \frac{1}{N} = .5$$

   i.e. $y = 0$

   As a result, we have

   $$y = \begin{cases} -x \\ 0 \\ x \end{cases}$$

   i.e. $y = |x|$.

4.5. MATLAB functions of the individual component and complete Architecture

4.5.1 LFSR

```
function q = lfsr(seed)
   n = length(seed);
   q = seed;
   switch n
      case 2
         q = [xor(q(1), q(2)) q(1:1)];
      case 3
         q = [xor(q(2), q(3)) q(1:2)];
      case 4
         q = [xor(q(3), q(4)) q(1:3)];
      case 5
         q = [xor(q(3), q(5)) q(1:4)];
      case 6
         q = [xor(q(5), q(6)) q(1:5)];
      case 7
         q = [xor(q(6), q(7)) q(1:6)]; %6,7
      case 8
         q = [xor(q(4),xor(q(5),xor(q(6), q(8)))) q(1:7)]; %4,5,6,8 (B8)
      case 9
         q = [xor(q(5), q(9)) q(1:8)];
      case 10
         q = [xor(q(7), q(10)) q(1:9)];
      case 11
         q = [xor(q(9), q(11)) q(1:10)];
      case 15
         q = [xor(q(14), q(15)) q(1:14)];
      case 17
         q = [xor(q(14), q(17)) q(1:16)];
      case 18
         q = [xor(q(11), q(18)) q(1:17)];
      otherwise
         error('Seed length exceeded or not supported!')
   end
end
```

4.5.2. SNG

```
function Y = sngup(p, n, s)
    if(nargin<2)
        n=10;
    end
    if(nargin<3)
        s = randi([0 1], 1, n);
    end
    L = 2^n-1;
    Y = zeros(1, L);
    A = fliplr(double(dec2binvec(p*L, n)));
    for i=1:L
        X = lfsr(s);
        Y(i) = bincmp(A, X);
        s = X;
    end
end
```

4.5.3. DRU

```
function y = druup(X)
    y=0; L=length(X);
    for i=1:L
        if(X(i)==1) y=y+1; end
    end
    y=y/L;
end
```

4.5.4. Binary comparator

```
function Y = bincmp(A, X)
    k = 1;
    n = length(A);
    while(k<=n && A(k)==X(k))
        k = k + 1;
    end
    k = k - 1;
    if(k==n)
        Y = 1;
    elseif(k<n && A(k+1)>X(k+1))
        Y = 1;
    else
        Y = 0;
    end
end
```

4.5.5. Scaled Addition

```
function Z = sadd(X, Y, W)
    if(nargin<3)
        W=sngup(0.5);
    end
    Z = or(and(not(W), X), and(W, Y));
End
```

4.5.6 Stochastic Absolute value

```
function out = sabs(X)
    numStates = 32;
    SAbsStates  = [1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0, 0 1 0 
1 0 1 0 1 0 1 0 1 0 1 0 1];

    currentState = 1;
    L = length(X);
    out = zeros(1, L);
    for i=1:L
        if(X(i)==0 && currentState>1)
            currentState = currentState - 1;
        elseif(X(i)==1 && currentState<numStates)
            currentState = currentState + 1;
        end
        out(i) = SAbsStates(currentState);
    end
end
```

4.5.7. Robert's Edge Detection

```
% y(i,j) = 0.5*(|x(i,j) - x(i+1,j+1)| + |x(i,j+1) - 
x(i+1,j)|);

clc; clear all; close all;
tic;

% x = imread('lens.tif');
x = [zeros(80,10), ones(80,10)*255,zeros(80,10), 
ones(80,10)*255,zeros(80,10), 
ones(80,10)*255,zeros(80,10), ones(80,10)*255];  % 
vertical edge
% x = x';                                   % horizontal edge
 %x = tril(ones(10,10));                    % diagonal edge
% x = rot90(x);                             % the other diagonal 
edge
```

```
[m,n] = size(x);

%padding for processing using robert's cross operator
x1 = zeros(m+1,n+1);
x1(1:m,1:n) = x;
x1(m+1,n+1) = x(m,n);
x1(m+1,1:n) = x1(m,1:n);
x1(1:m,n+1) = x1(1:m,n);
toc;

y_conv = zeros(size(x));
for i=1:m
    for j=1:n
        y_conv(i,j) = 0.5*(abs(x1(i,j) - x1(i+1,j+1)) +
abs(x1(i,j+1) - x1(i+1,j)));
    end
end
toc;

x1 = x1/max(max(x1));
y = zeros(size(x));
for i=1:m
    for j=1:n
        a = sngup(x1(i,j));
        b = sngup(x1(i+1,j+1));
        c = sngup(x1(i,j+1));
        d = sngup(x1(i+1,j));
        y(i,j) = druup(sadd(sabs(sadd(a, not(b))),
sabs(sadd(c, not(d)))));
    end
end
% post-proccessing
 threshold = 0.65; % other values 0.7, 0.8
 for i=1:m
    for j=1:n
        if(y(i,j)>threshold) y(i,j)=1;
         else y(i,j)=0;
         end
     end
 end
toc;

figure;
subplot(1,3,1), imshow(x, []); title('Original');
subplot(1,3,2), imshow(y_conv, []);
title('Conventional');
subplot(1,3,3), imshow(y, []); title('Stochastic');
```

4.6. Making of MACROS by Circuit-maker Software

MACROS

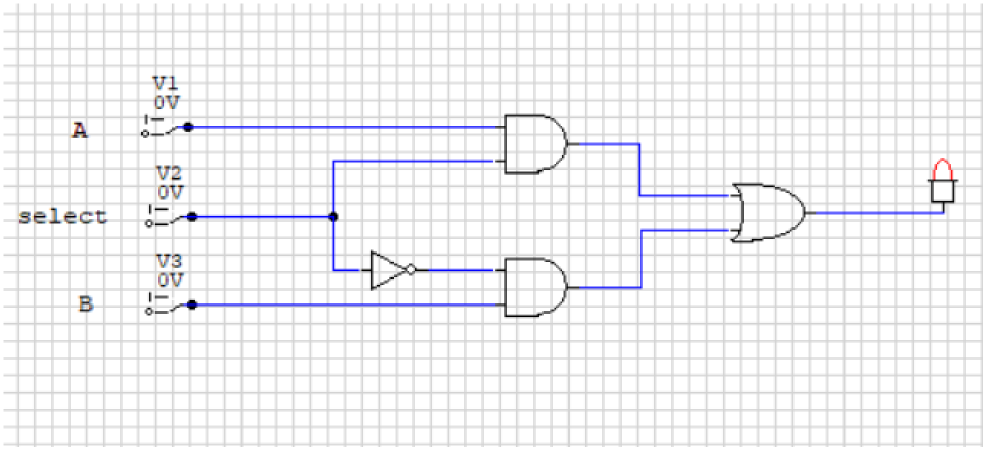
STEPS

1. Design your circuit.
2. Run and check the circuit.
3. Remove all the switch from input terminals.
4. Go to the “Macros”→→ “New macro”
5. Include present circuit inside new symbol→ “yes”
6. Give a macro name.
7. Go to “pins per package” and provide the no of required pins → “add package”.
8. Edit or delete the pins according to your needs, by clicking the pins in the chip.
9. After completion of editing→ “OK”
10. Connects all the inputs and outputs to the input and output terminals of the Macros respectively.
11. Go to the “Macros” → “save macros”
12. Provide MAJOR and MINOR class
13. Save macros.

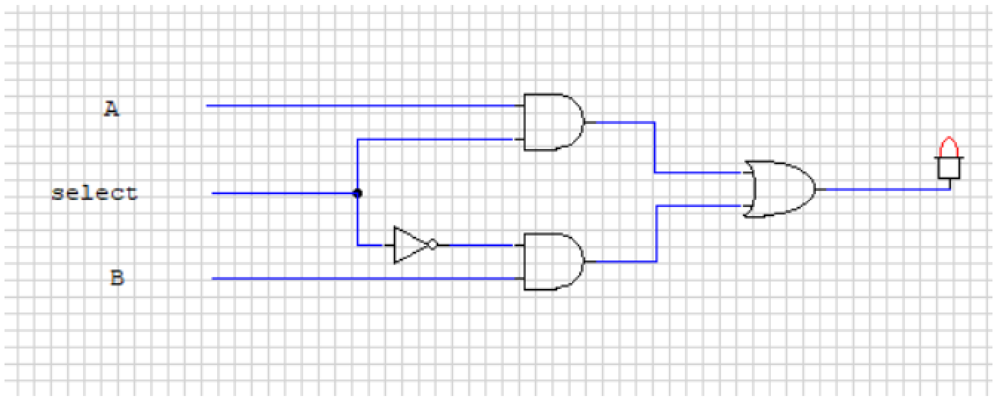
EXAMPLE

Design a 2:1 MUX:

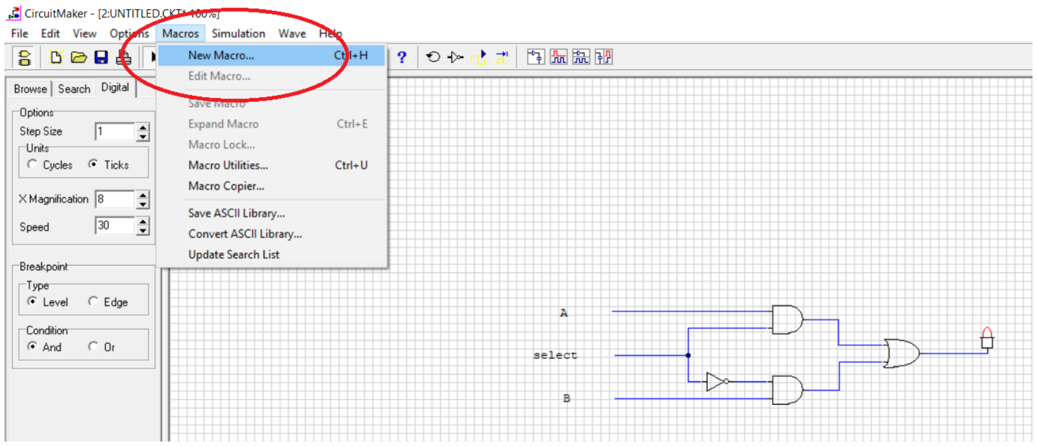
- CKT



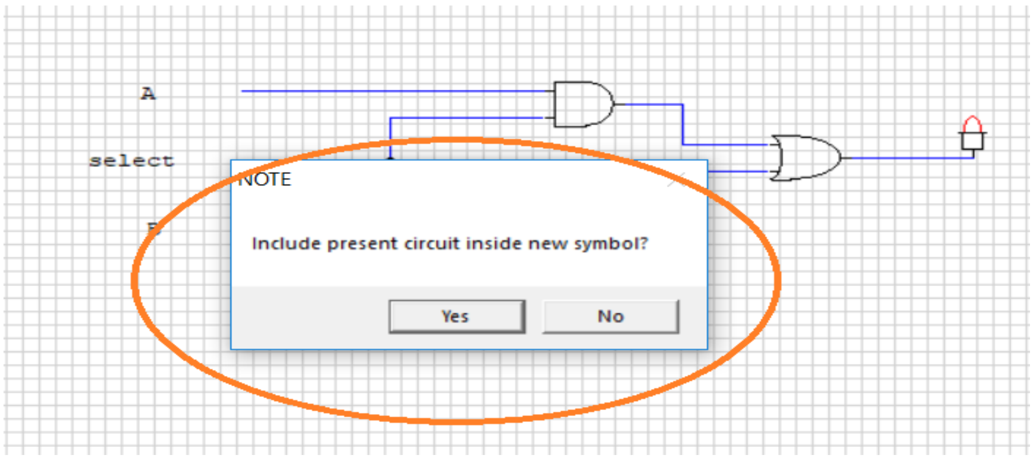
➢ Remove all the switch from input terminals.



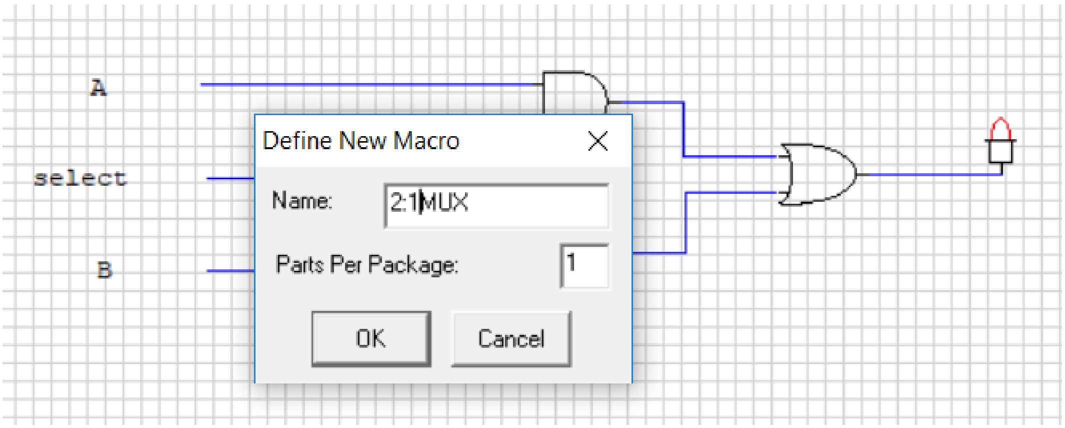
➢ Go to the "Macros"→→ "New macro"



➢ Include present ckt inside new symbol→ "yes"



➢ Give a macro name and click → "OK"



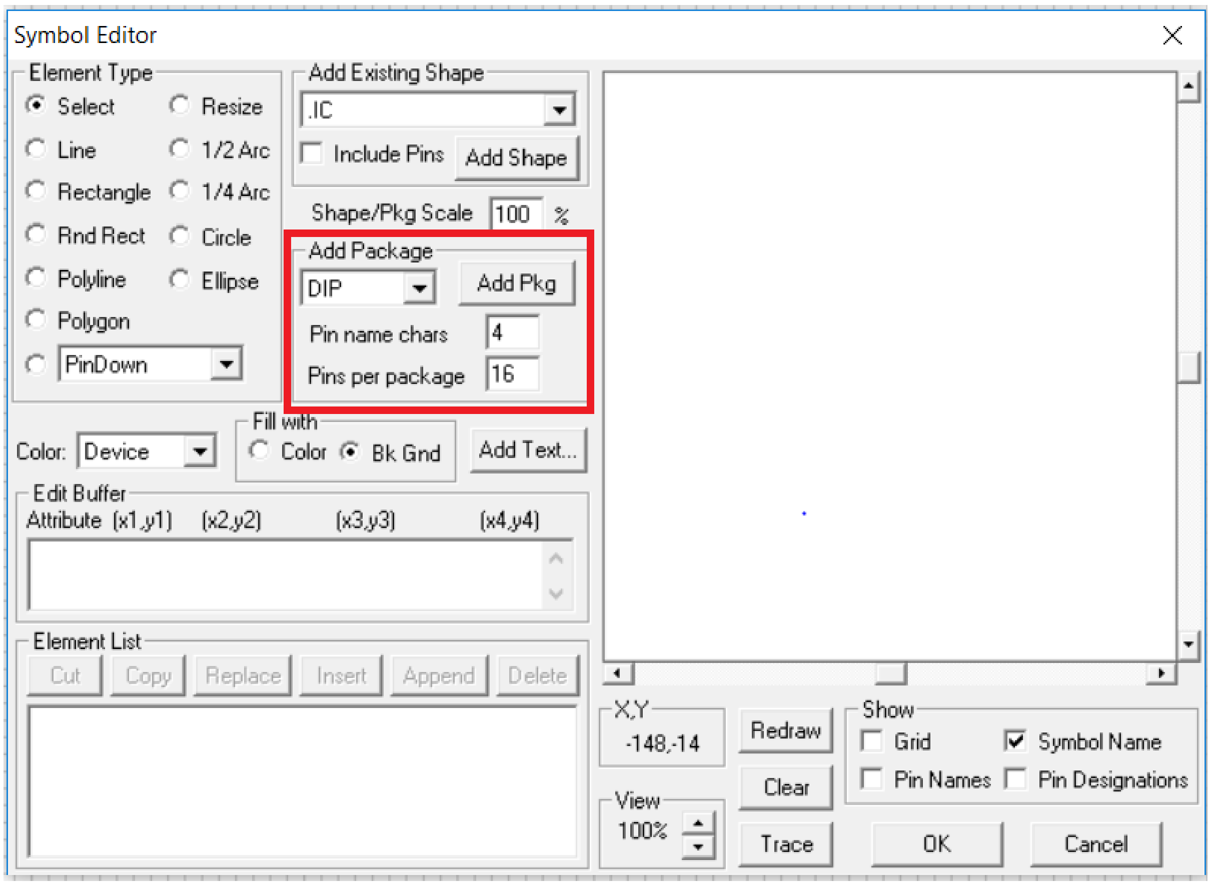
Define New Macro

Name: 2:1MUX

Parts Per Package: 1

OK Cancel

➢ Go to "pins per package" and provide the no of required pins → "add package".



Symbol Editor

Element Type: [✓ Select] Resize, Line, 1/2 Arc, Rectangle, 1/4 Arc, Rnd Rect, Circle, Polyline, Ellipse, Polygon, PinDown

Add Existing Shape: .IC; Include Pins [ ]; Add Shape

Shape/Pkg Scale: 100 %

Add Package: DIP; Add Pkg

Pin name chars: 4

Pins per package: 16

Color: Device

Fill with: Color [✓ Bk Gnd]

Add Text...

Edit Buffer: Attribute (x1,y1) (x2,y2) (x3,y3) (x4,y4)

Element List: Cut Copy Replace Insert Append Delete

X,Y: -148,-14

View: 100%

Redraw Clear Trace

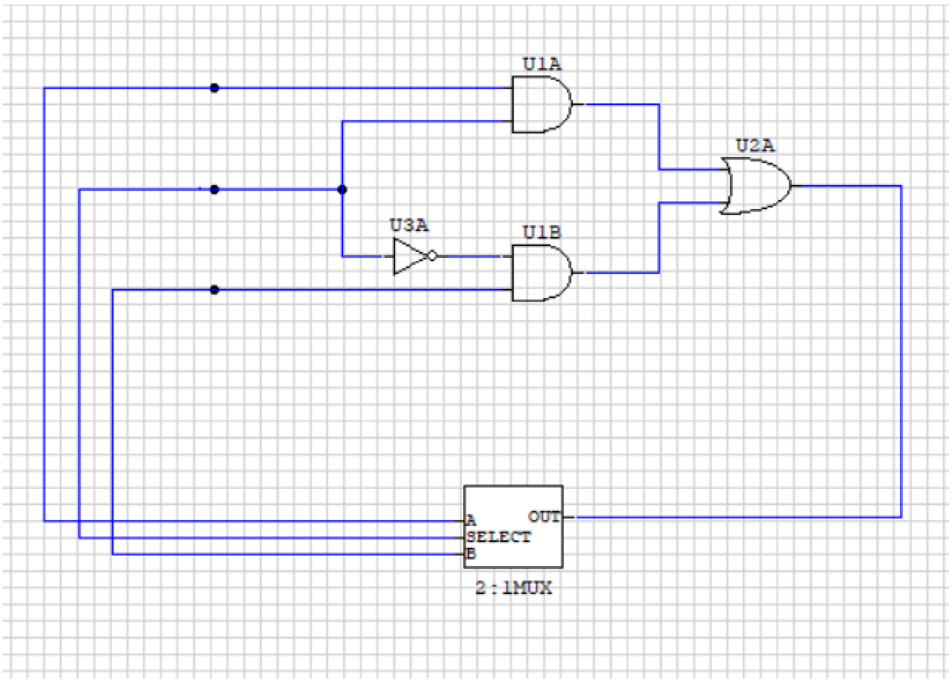
Show: Grid [ ] [✓ Symbol Name] Pin Names [ ] Pin Designations [ ]

OK Cancel

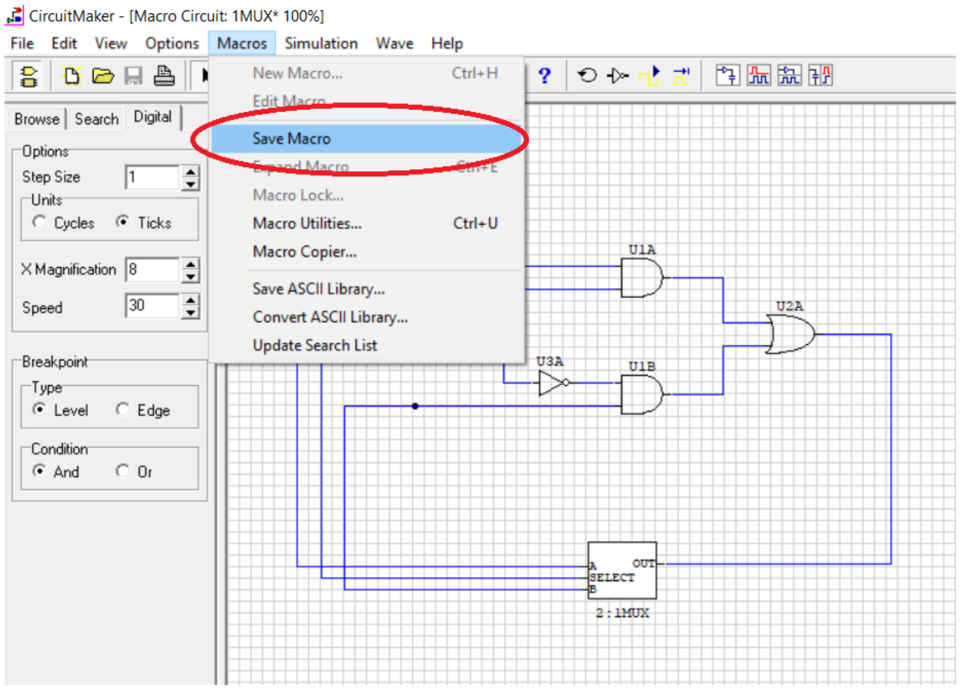
So for 2:1 MUX we need three input and one output. Macros are symmetric in nature. So we have to provide "pins per package = 6". It will create three pins each side. Then click on the "add package". It will create the macro. Clicking on the pins we can edit or delete pins further.

For this case 2 out of 3 pins from the right side (output side) will be deleted.

- After completion of editing→ "OK".
- Connects all the inputs and outputs to the input and output terminals of the Macros respectively.



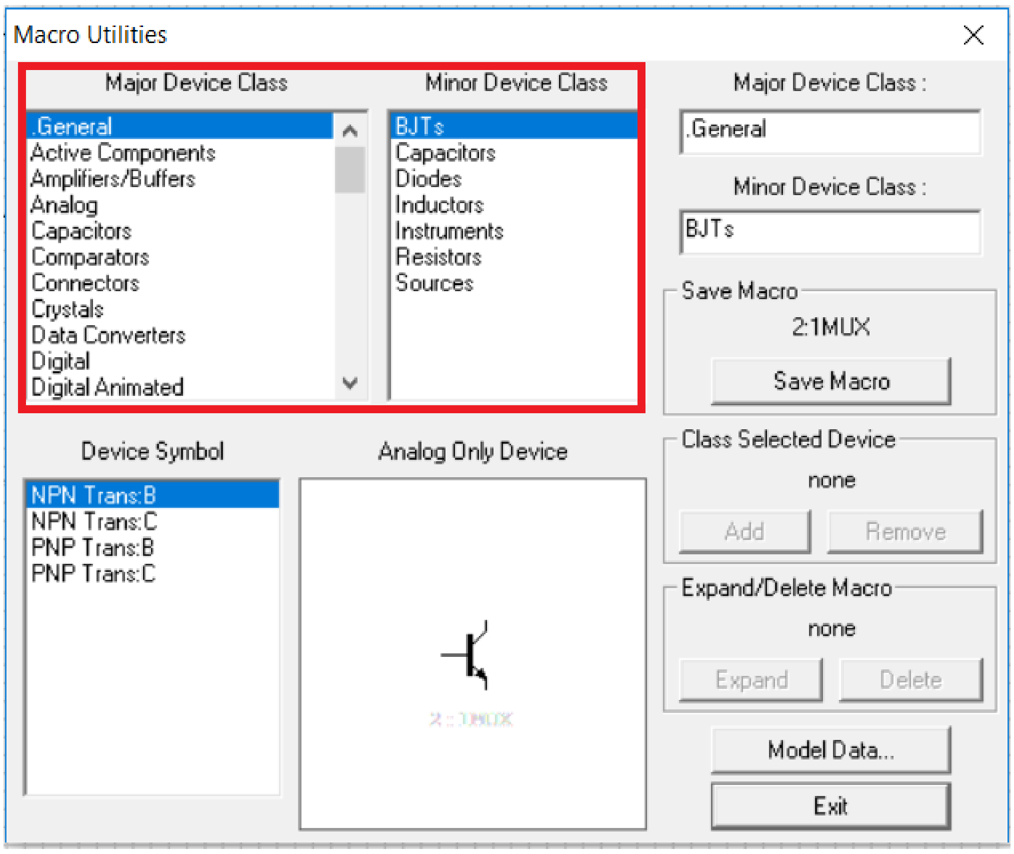
- Go to the "Macros" → "save macros".


CircuitMaker - [Macro Circuit: 1MUX* 100%]

Options
Step Size 1
Units: Cycles [✓ Ticks]
X Magnification 8
Speed 30
Breakpoint
Type: [✓ Level] Edge
Condition: [✓ And] Or

- Provide MAJOR and MINOR class.
  You may choose major and minor class from existing list or create for your own.

- For existing :
  Choose your major and minor class from the list of Red box respectively.



Major Device Class: .General, Active Components, Amplifiers/Buffers, Analog, Capacitors, Comparators, Connectors, Crystals, Data Converters, Digital, Digital Animated

Minor Device Class: BJTs, Capacitors, Diodes, Inductors, Instruments, Resistors, Sources

Major Device Class : .General

Minor Device Class : BJTs

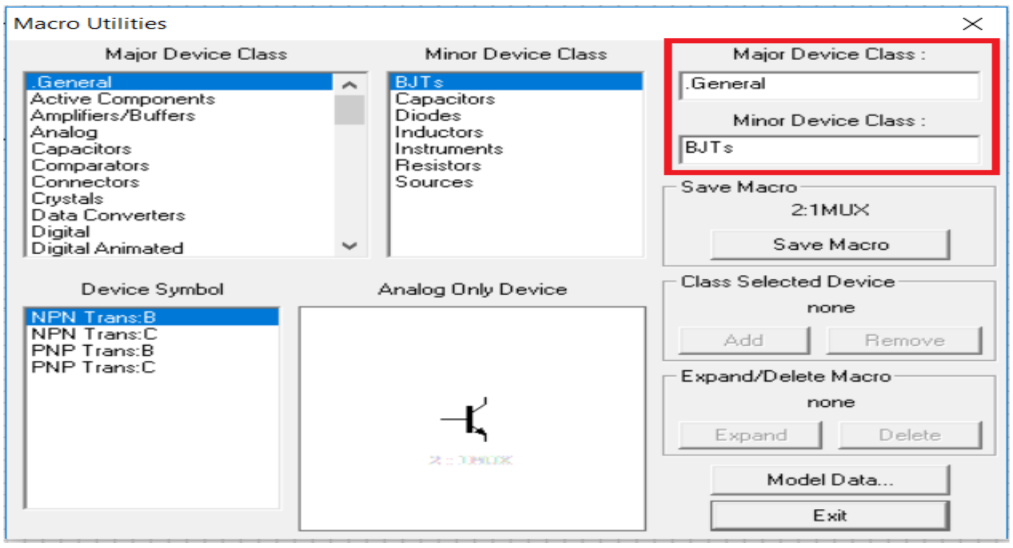
Save Macro: 2:1MUX

Class Selected Device: none

Expand/Delete Macro: none

Device Symbol: NPN Trans:B, NPN Trans:C, PNP Trans:B, PNP Trans:C

- For creation:
  Provide major and minor class name in the Red box in the respective places.



Major Device Class: .General, Active Components, Amplifiers/Buffers, Analog, Capacitors, Comparators, Connectors, Crystals, Data Converters, Digital, Digital Animated

Minor Device Class: BJTs, Capacitors, Diodes, Inductors, Instruments, Resistors, Sources

Major Device Class : .General

Minor Device Class : BJTs

Save Macro: 2:1MUX

Class Selected Device: none

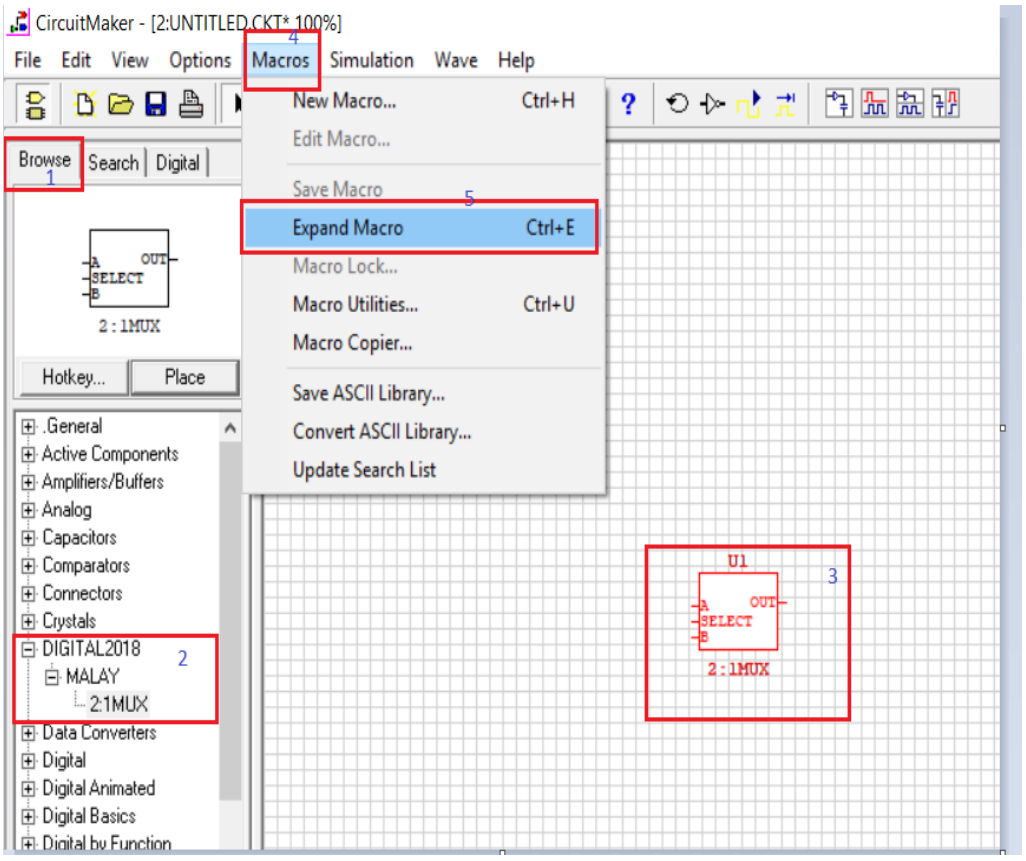
Expand/Delete Macro: none

Device Symbol: NPN Trans:B, NPN Trans:C, PNP Trans:B, PNP Trans:C

➢ Then go to "Save Macro". Your macro will be saved in your specified class.

➢ You can see or edit the internal ckt of the macro by expanding the macro.

- "Browse"→ "major class name"→ "minor class name"→ "macro name" → place it→ select macro → "Macros"→ "Expand macro".



All this Macros are very helpful, when we try to design the any circuit which is very large. In the stochastic computing this macros has been used to complete design the circuit. It is easier to understand the circuit and easy to design the whole circuit. Macros of LFSR and SNG has been shown in the earlier stage of this chapter.

Chapter 5
Results and Discussions

Results and Discussions

There are lots of application of stochastic computing [25]. Stochastic computing can be used for analysis of reliability [26], neural network computation [3], other operation of control system [22], besides the creation of polynomial, ordinary arithmetic operation. These are the basic application of the stochastic computing. We can use stochastic computing in the field of Image processing and stochastic computing have very high potential in that section. In this chapter we will discuss about the application of the stochastic computing in the implementation of the elements or application of the image processing.

5.1. Robert's Edge Detector

Edge detection is the most frequently used method for segmenting images based on abrupt changes in intensity. In the image processing, edge detection is one of the essential weapon or tool which tells us the location of abrupt change or sudden change of the intensity level in the image and it creates an edge at that location of discontinuity of brightness. There are many number of edge detection operators available, each are introduced to be sensitive to certain types of edges. The conventional method of edge detection introduce convolution of an image with a filter, which is constructed to be sensitive to large gradients in the image while returning values of zero in the uniform regions[8]. All this operator can be constructed by stochastic computing elements discussed in the 3rd chapter. Here we are using Robert's cross operators .Robert's cross operators are one of the earliest method using 2-D masks with a diagonal preference. The Roberts operators are based on implementing the diagonal differences.

Let $x_{i,j}$ denotes the pixel value at location (i,j) of the original image and $y_{i,j}$ is the pixel intensity value at location (i,j). the convolution between the original image and the Robert's cross operator and produce output as given as below

$$y_{i,j} = \frac{1}{2}[|x_{i+1,j+1} - x_{i,j}| + |x_{i,j+1} - x_{i+1,j}|]$$

From the above equation we can say we need stochastic element such as scaled addition, stochastic absolute value, all of them are discussed in the previous chapter.

$$\begin{array}{|c|c|} \hline +1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad \qquad \begin{array}{|c|c|} \hline 0 & +1 \\ \hline -1 & 0 \\ \hline \end{array}$$

$$G_X \qquad \qquad \qquad \qquad G_Y$$

fig 5.1: Robert's cross operator

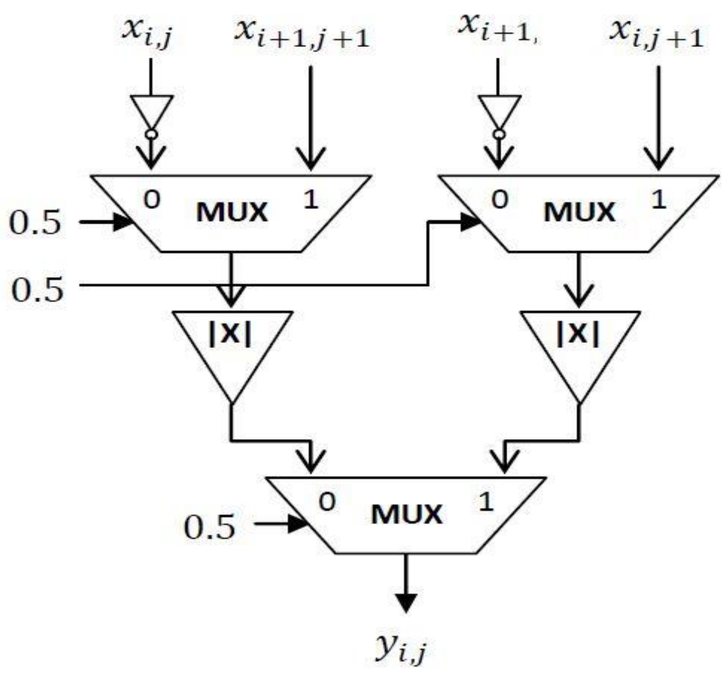


fig 5.2: Robert's Edge Detector Architecture using Stochastic element

5.1.1. Results

5.1.1.1. With Noiseless Image

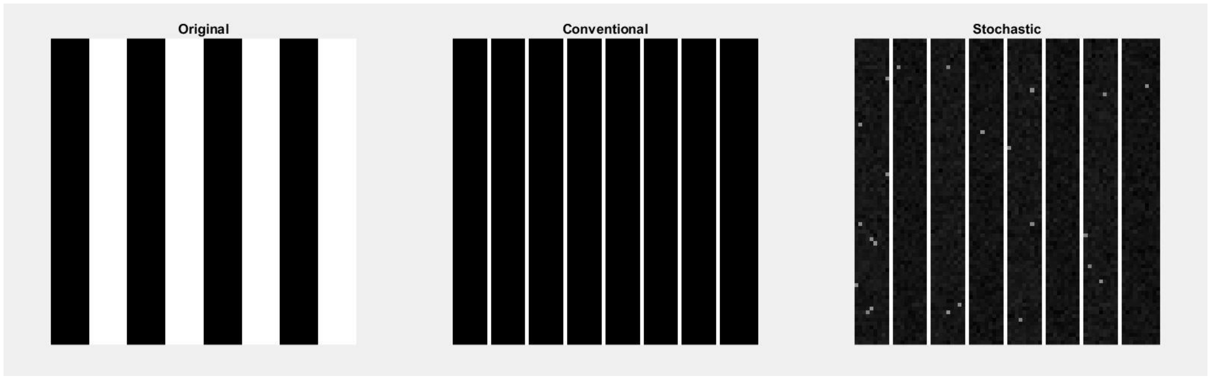


fig 5.3: output of Robert's Edge Detector with Noiseless Image

5.1.1.2. With Noisy Image

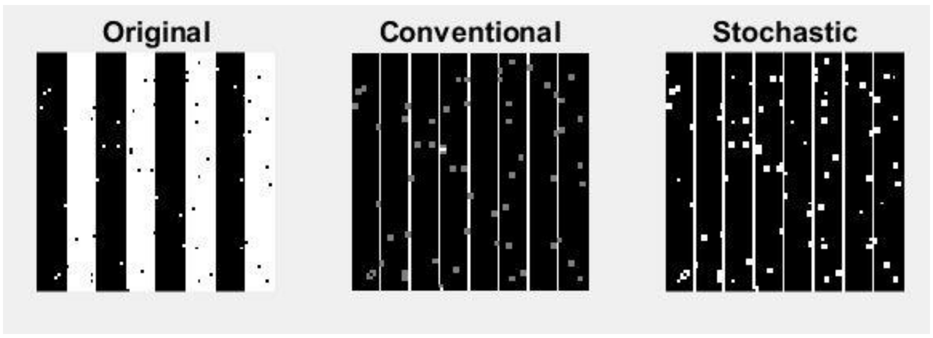


fig 5.4: Output of Robert's Edge Detector with Noisy Image

5.1.2. Discussion

From the above result we can say that, the output from the both conventional and stochastic approach is almost same in the both case whether we use noiseless or noisy image. But the advantage of the stochastic approach is it requires very less no of hardware and its very low cost implementation technique. So we have seen that we can create a stochastic Robert’s Edge detector which can be used for edge detection purpose and its output may be accepted under any circumstances. Also we have seen that Robert’s Edge detector is not suitable for the noise reduction purpose. We know that for the noise reduction purpose Sobel operator is more suitable along with the edge detection. The disadvantage of the stochastic approach over the conventional method is, stochastic computing takes very long time to compute.

5.2. Noise reduction

Noise reduction may be another application of the stochastic computing in the image processing. Median filter is the well-known order statistic filter. When it is applied to an image, after convolution between the original image and the median filter, the median of the intensity value in the neighbourhood of that image replaces the centred value. As median filter gives super noise- reduction capabilities, with very less blurring than other similar size smoothing filter [8], median filter is very much popular.

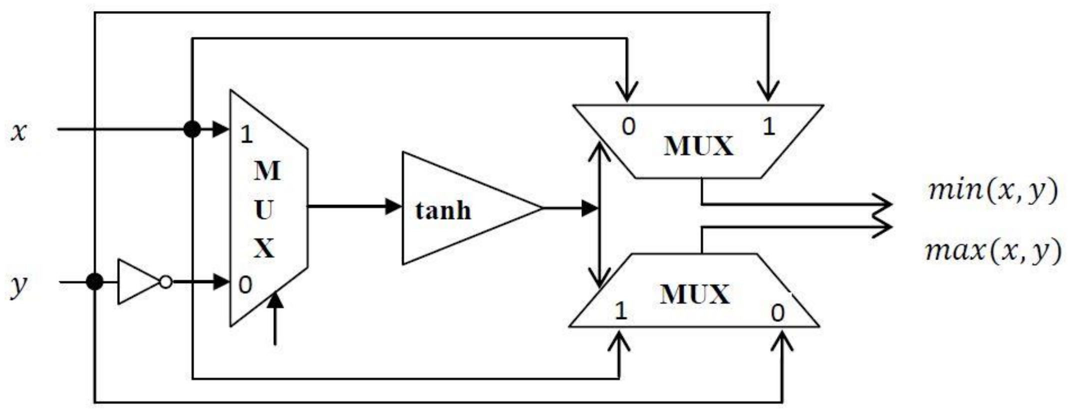


fig 5.5: Architecture for Noise reduction

5.2.1. Results

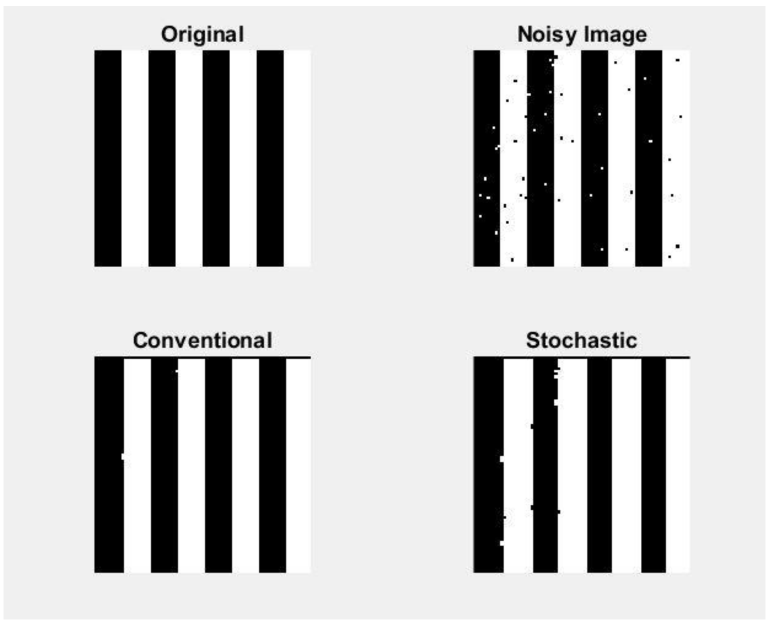


fig 5.6:Resullt of Noise reduction and comparison

5.2.2. Discussion

From the above result we can say that stochastic logic can be used in the image processing application for the noise reduction and this produce the faithful outcomes as it is almost similar to the conventional method but requires very less no of hardware. This result will be more accurate if we use larger no of bit for representation of a value, as a result bit stream will have more length so, so accuracy will be better, but for the increase in bit, stochastic logic will take more and more no of clock pulse so time consumption of the circuit will be more. To reduce the time consumption we may use Pulse Width Modulation signal to represent the probability value, it will reduce the time consumption significantly.

5.3. Unsharp masking

Unsharp masking[21] is the method that is used for long time by printing and publishing industry to sharp images consists of subtracting an unsharp (smoothed) version of image from the original image. This is known as unsharp masking, consists of the following steps.

1. Blurring the actual image
2. Subtraction of blurred image from the actual image (the result produce mask).
3. Addition of mask to the actual image.

Let b(x,y) represents the blurred image and f(x,y) represents the actual image, so we can write the equation of the mask is

$$s_{mask}(x,y) = f(x,y) - b(x,y)$$

Now, after addition a weighted mask to the actual image

$$s(x,y) = f(x,y) - k * s_{mask}(x,y)$$

Where $s(x,y)$ is the final image. The introduction of of weight 'k' is for generic purpose. When k=1, we get unsharp masking, as mentioned above. If k>1, then it is highboost filtering, if k<1 de-emphasizes the contribution of the unsharped masking.

5.2. Results

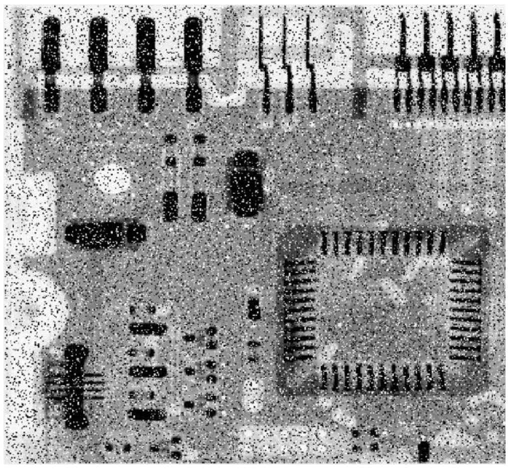


fig 5.7: Original Image

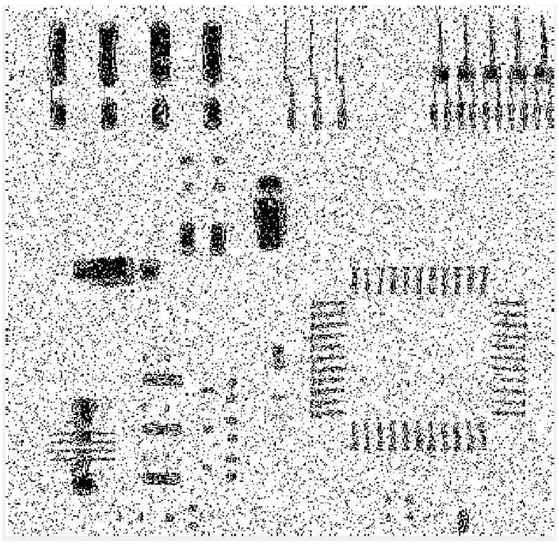


fig 5.8: High boost image using stochastic computing

5.2. Discussion

In this experiment we have seen that stochastic computing may be used for boosting any image so that contrast may be increased. It helps to boost the higher intensity value to be more and lower intensity may be more less than original so brightness of the image will be more. So contrast enhancement is also possible for the stochastic computing.

Chapter 6

Conclusion and Future Scope

Conclusion and Future Scope

6.1. Conclusion

In 1960s, stochastic computing was introduced firstly as a combination of digital-analog computing technique that gives designers a confidence to implement complex arithmetic function using very less no of hardware. But later transistor became very cheaper and designers were attracted to the conventional system due to its fast response, even conventional method is less effective in chip area and cost. Due to this stochastic computing had very few application area. But now after development of technology, new challenge is to reduce the hardware, chip area and energy consumption for the circuit. The stochastic computing system can be used to solve this type of problem faced by the conventional system with the high accuracy. This work shows the performance of the stochastic computing system in the digital implementation of the various elements using both sequential and combinational construction. This work is mainly on the implementation of the various elements and their use in the image processing application using the stochastic computing. The main purpose of using the stochastic computing is very low hardware requirement and highly error tolerate capabilities. In the different application stochastic computing is used successfully, and giving us confidence that it may be a right choice for the replacement of the conventional system. The main problem of the stochastic computing is very high computational time and if we want high accuracy then it needs very long time. If we increase no of bit for representation of probability by one, then the no it doubles the no of randomly generated number, so execution needs more time. In this thesis many things are implemented and results show that accuracy is good enough as compare to the conventional method. Still many research is running on the Stochastic Computing so that in the coming days all the drawback of the stochastic computing can be avoided.

6.2. Future Scope

1. A potential area of area of the stochastic computing is, use of the PWM technique and the parallel implementation for the fast response and reduce the long computational time.
2. Further study on the correlated input, may be one of the greatest work because, for correlated input accuracy is a serious issue.
3. Study on storing of randomly generated stochastic no is also a biggest issue for a large system, as intermediate results may be useful for any stage of the execution. Some work must be done on this issue.
4. Design and implementation of advanced application by the stochastic computing.

References

[1] B.R. Gaines, "Stochastic Computing Systems, Advances in Information Systems Science," J.F.Tou, ed., vol.2, chapter 2, pp.37-172, New York: Plenum, 1969.

[2] H. Chen, J. Han, and F. Lombardi, —A transistor-level stochastic approach for evaluating the reliability of digital nanometric CMOS circuits,|| in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst., Oct. 2011, pp. 60–67.

[3] B. D. Brown and H. C. Card, "Stochastic neural computation. I.computational elements," IEEE Transactions on Computers, vol. 50, no. 9, pp 891-905, 2001.

[4] S. S. Tehrani, S. Mannor, and W. J. Gross, "Fully parallel stochastic ldpc decoders," IEEE Transactions on Signal Processing, vol. 56, no. 11, pp. 5692–5703, 2008.

[5] S. S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. J. Gross, "Majority-based tracking forecast memories for stochastic ldpc decoding," IEEE Transactions on Signal Processing, vol. 58, no. 9, pp. 4883–4896, 2010.

[6] Y. Liu and K. K. Parhi, "Lattice FIR digital filters using stochastic computing," in Proceedings of 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015.

[7] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," IEEE Trans. Device and Materials Reliability, 5, 3, pp. 305–316, 2005. doi:10.1109/TDMR.2005.853449.

[8]R. C. Gonzalez and R. E. Woods, —Digital Image Processing,|| second Ed., Pearson, 2008.

[9] Umbaugh, Scott E (2010). Digital image processing and analysis : human and computer vision applications with CVIPtools (2nd ed.). Boca Raton.
[10]W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja, —An architecture for fault-tolerant computation with stochastic logic||, IEEE Transactions on Computers, Vol. 60, No. 1, January 2011.

[11] Peng Li, David J. Lilja, Weikang Qian, Marc D. Riedel and Kia Bazargan, "Logical Computation on Stochastic Bit Streams with Linear Finite State Machines," IEEE Transactions on Computers, Volume: 63, Issue: 6, June 2014.

[12] Peng Li, David J. Lilja, Weikang Qian, Marc D. Riedel and Kia Bazargan, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2012.

[13] P. Li and D. Lilja, —Using stochastic computing to implement digital image processing algorithms,|| in 29th IEEE Int. Conf. Computer Design (ICCD), Oct. 2011, pp. 154-161.

[14] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, ―Computation on stochastic bit streams digital image processing case studies,‖ IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 3, pp. 449–462, Mar. 2014.

[15]M. H. Najafi, S. Jamali-Zaraveh, D. Lilja, M. Riedel, K. Bazargan and R. Harjani, ―Time-Encoded Values for Highly Efficient Stochastic Circuits‖, IEEE Transactions on Very Large Scale integration (VLSI) Systems, vol. 25, pp. 1644-1657, January 2017.

[16] M. Hassan Najafi and David J. Lilja, "High-Speed Stochastic Circuits Using Synchronous Analog Pulses," 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), 2017.

[17] Young, Tzay Y., King-Sun Fu. Handbook of Pattern Recognition and Image Processing. Academic Press, San Diego, California, 1986.

[18] Russ, John C. The Image Processing Handbook. CRC Press, Boca Raton, Florida, 1995.

[19] Edges: The Canny Edge Detector, Http:// www.icbl.hw.ac.uk/marble/vision/low/ edges/canny.htm.

[20] Basic Image Processing Demos, Http:// robotics.eecs.berkelee.edu/~mayi/ imgproc/

[21] W. Qian M. D. Riedel and I. Rosenberg, ―Uniform approximation and Bernstein polynomials with coefficients in the unit interval,‖ Eur. J. Combinatorics, vol. 32, pp. 448-463, 2011.

[22] A. Dinu, M. N. Cirstea, and M. McCormick, "Stochastic implementation of motor controllers," In Proceedings of the IEEE Symposium on Industrial Electronics. 639–644, 2002.
[23] Peng Li, David J. Lilja, Weikang Qian, Marc D. Riedel and Kia Bazargan, "Logical Computation on Stochastic Bit Streams with Linear Finite State Machines," IEEE Transactions on Computers, Volume: 63, Issue: 6, June 2014.

[24] Peng Li, David J. Lilja, Weikang Qian, Marc D. Riedel and Kia Bazargan, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2012.

[25] Armin Alaghi and John P. Hayes, ―Survey of stochastic computing,‖ ACM Trans. Embed. Comput. Syst., 12(2s):92:1–92:19, May 2013.

[26] H. Aliee, and H. R. Zarandi, "Fault tree analysis using stochastic logic: A reliable and high speed computing," In Proceedings of the Reliability and Maintainability Symposium, 1–6, 2011.