2014

VLSI ARCHITECTURES FOR MEAN-SHIFT BASED OBJECT TRACKING

Ruby Mishra
Department of Electronics and Communication Engineering
National Institute Of Technology, Rourkela

VLSI ARCHITECTURES FOR MEAN-SHIFT BASED OBJECT TRACKING

A thesis submitted in the partial fulfillment of the requirements for the

degree of

Master of Technology

in

VLSI DESIGN AND EMBEDDED SYSTEMS

Submitted by

Ruby Mishra

(Roll No: 212EC2471)

Under the guidance of

Prof. Kamala Kanta Mahapatra

Professor

Department of Electronics and Communication Engineering
National Institute Of Technology
Rourkela

May2014

To my family and teachers

Department of Electronics & Communication Engineering
NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA
ODISHA, INDIA – 769 008

CERTIFICATE

This is to certify that the thesis titled "**VLSI Architectures for Mean-Shift based Object Tracking"** submitted to the National Institute of Technology, Rourkela by **Ms. Ruby Mishra**, Roll No. **212EC2471** for the award of the degree of **Master of Technology** in **Electronics & Communication Engineering** with specialization in **"VLSI Design and Embedded Systems"**, is a bonafide record of research work carried out by her under my supervision and guidance. The candidate has fulfilled all the prescribed requirements.

The thesis, which is based on candidate's own work, has not been submitted elsewhere for a degree/diploma. To the best of my knowledge, the thesis is of standard required for the award of a Master of Technology degree in Electronics & Communication Engineering.

Date:02-05-2014

Place: Rourkela

Prof. K. K. Mahapatra
Department of Electronics & Communication Engineering
NATIONAL INSTITUTE OF TECHNOLOGY
Rourkela-769 008 (INDIA)

ACKNOWLEDGEMENT

I am very much grateful to my thesis guide **Prof. K.K. Mahapatra** for his guidance, advice, encouragement and support throughout my thesis work. I am indebted to him for helping me to learn the research and writing skills, which have been very beneficial for current research and will also be for my future career. Without his efforts and patience this research would have never been possible to complete. The concepts, techniques and results presented in this thesis are being guided by him in one or the other way. It has been a great honour and pleasure for me to do research under supervision of Prof. K.K. Mahapatra. I would like to thank him for being my advisor here at National Institute of Technology, Rourkela.

Next, I want to express my sincere thanks to **Prof. D.P. Acharya, Prof. Ayas Kanta Swain and Prof. Alok Satpathy** for their continuous moral support and technical help from time to time without which it would have been very difficult for completing this research project.

I also extend my respects to **Prof. S. Meher, Prof. S. K. Patra, Prof. N. Islam, Prof. Pramod K. Tiwari, Prof. Samit Ari, Prof. Poonam Singh** for teaching me and also helping me to acquire the learning skills.

Besides, my sincere thanks to Mr. Vijay Kumar Sharma, Ph.D Scholar ECE dept. for his technical support for the completion of my project. It is he who has helped me to learn the programming and optimizing skills for which I would be grateful to him throughout my life.

I thank all my faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela, for their kind help for completing this thesis.

I would also like to thank all my friends for their co-operation and inspiration at every moment I needed. I also thank some special persons like Kanhu Sir, Venkat Sir, Rajesh Sir, Sudheendra Sir, Tom Sir and Jagannath Sir for their continuous support in each and every situation during my project period.

Finally my parents, who are Almighty in disguise. I am indebted for their love, sacrifice, and support throughout their life. It is only because of their blessings and encouragement, today I am able to withstand in every adverse situation in my life.

RUBY MISHRA

Contents

List of Figures

List of Tables

Abstract

The demand for real-time video surveillance systems is increasing rapidly. The purpose of these systems includes surveillance as well as monitoring and controlling the events. Today there are several real-time computer vision applications based on image understanding which emulate the human vision and intelligence. These machines include object tracking as their primary task. Object tracking refers to estimating the trajectory of an object of interest in a video. A tracking system works on the principle of video processing algorithms. Video processing includes a huge amount of data to be processed and this fact dictates while implementing the algorithms on any hardware.

An efficient video processing algorithm is adopted here for estimating the trajectory of moving objects in a video. The tracking algorithm is based on mean-shift iteration technique. This method tracks accurately the target object in a sequence of video frames. The key objective is to implement the algorithm on an FPGA platform with less computational complexity and hardware utilization for real-time applications. Two VLSI architectures for the mean-shift based object tracking system are implemented and verified. The FPGA target device used here is XILINX xc5vlx110t.

The architectures consist of many divider modules which plays a significant role in the performance of the system. Divider includes shifting and addition operations repeatedly to get a particular result. Hence emphasis should be given for the design of an optimized divider unit. Here a serial divider using non-restoring algorithm is implemented in 90 nm technology using CADENCE tool.

Chapter 1

Object Tracking: an Overview

- Introduction
- Object and its representation
- Selection of object feature for tracking
- Detection of Object
- Object Tracking
- Literature Review
- Problem Description
- Objective
- Organisation of thesis
- Conclusions

1.1 Introduction

Real-time computer vision applications like airport safety, road traffic control, video surveillance, robotics, natural human-machine interface, etc. [1], [2] are of great importance today. These applications include machines that can visualize and understand their environment and react according to the perceived parameters and features [2]. This fact signifies the capability of these systems to detect and track objects. Object tracking is thus considered to be the basic task in these kinds of applications [1], [3]. It is a method of detecting the objects moving in a video with respect to time and pursue the objects of interest by estimating the motion parameters [2]. Motion parameters include trajectory, speed and orientation of the object to be tracked [1]. The tracking algorithm as well as the hardware used defines the efficiency of a good tracker. The tracker also provides information about the object while performing the tracking in several frames of a video. The information includes shape, area and position of the object. Hence its various application areas are traffic monitoring, video surveillance system, vehicle navigation, and weather monitoring.

Tracking a moving object is an active research area in the field of image processing and computer vision. Today object tracking is the key technology behind video tracking. Recent advancements in computer technology and development of high quality cameras have attracted many engineers to put interests to develop object tracking algorithm [4]. This chapter gives an overview of object tracking and its available methodologies.

The challenges arising in tracking is due to the change in object position continuously. Hence it gives rise to different issues in the field of object tracking. Some of these are, noise in an image, complex shape of objects, occlusions, changes in illumination of object, loss of 3d video information in 2D image and finally the requirements of real time processing of the entire tracking system[4].

A number of questions arise before approaching to a specific tracking algorithm. These are

1. How can an object be represented for tracking?
2. What are the features of the object in an image that can be taken for tracking?
3. How to model the shape and movement of the object?

This chapter gives brief knowledge to all these questions along with it discusses some of the methods used for object tracking.

1.2 Object and Its Representation

In tracking, object plays a vital role and it can be defined well by taking some examples of an object. Object can be a ship in a sea, a person moving in a road, vehicle passing in a road, missiles in the air etc. An object is defined to be anything of interest, for analysis [4]. The object is represented by its shape and appearance. The shape of an object can be represented by (i) Points (ii) Geometric shapes (iii) Object contour and (iv) Skeletal model etc.

For tracking small regions in an image, point representation is used. ***Point*** represents the centroid of the object. Sometimes multiple points in an image are also used to represent the shape of an object. In ***geometric*** shape representation, object is represented by ellipse and rectangular shapes. ***Contour*** representation concentrates on the boundary of an object. Silhouette of an object is considered to be a portion inside it. Geometric shapes can be used to represent both rigid and non-rigid objects. Contour representations are best suited for tracking non-rigid complex objects.

An appearance of an object can be represented as many parameters but two most important representations are as follows:

(i)Probability densities of an object (ii) Templates of an object.

The probability density estimation may be parametric like Gaussian or non-parametric representation such as Parzen windows. The probability densities are calculated by taking the region defined by a contour or an ellipse. There exists a relationship between object tracking and object representation [4].

1.3 Selection of Object Feature for Tracking

Once the object is represented in a tracking system the next important issue is to select an appropriate feature of object for tracking. The different types of features for tracking are (i)Color (ii) Edges and (iii) Texture.

Color is an important feature mostly used in histogram based object representation. The two important factors which influence the object color are: the surface reflectance property and the illuminant spectral power density. Color is represented by three colors i.e.

red, green and blue in RGB color space but HSV color space which represents hue, saturation and value is more preferred.

Edge is used to detect the boundary of an object. This usually plays a significant role in evaluating the image intensities. The edges are not much sensitive to changes in illumination as compared to color. Hence this is a simple and more accurate method which is used in the places where boundary of the objects is to be tracked.

Texture represents the properties of objects such as regularity and smoothness. It represents the intensity variation of the object surface. Texture is also less sensitive to changes in illumination.

Automatic feature selection is now-a-days gaining popularity. The various selection schemes are filter method and wrapper methods [4].Color being the mostly used feature uses color histogram to represent object appearance but color is sensitive to illumination variations.

1.4 Detection of Object

Detection of an object is essential in tracking. There are four commonly used object detection techniques: (i) point detectors (ii) Segmentation (iii) background histogram [4]. Object detection uses the information in a single frame to track the object. There may be some errors in single frame, hence information from multiple sequence of frames are used to avoid noisy detection. This helps in differentiating the changing regions in the frames and then tracker observes the similarity between the frames to perform the successful tracking.

Point detectors use the concept of interest points in an image to do the tracking. Interest points are independent of the intensity of illumination. Out of many available interest point detectors one of the important detectors is **SHIFT** detector. SHIFT is abbreviated as Scale Invariant Feature Transform and was introduced by Lowe in 2004. The basic steps of the SHIFT are:

i. Gaussian filter is convolved with the image to construct a scale space.
ii. A Gaussian difference image is generated by using convolved images.
iii. Interest points set are used to select maxima and minima of the Gaussian difference image.

iv. Update of each candidate location is done by interpolating the color from the neighboring pixels.

v. Then the candidates along edges or those having low contrast are eliminated.

vi. The rest of the interest points orientations are assigned in a small neighborhood of the candidate point depending on the peak values in the histograms of gradient directions [4].

In **Segmentation,** the image is partitioned into smaller regions [4]. Hence a good partitioning method should be used for successful object tracking. Mean-shift approach is used for segmentation of an image. The segmentation algorithm is as follows:

i. A number of cluster centers are created from an image data.

ii. Each cluster center is moved to the mean value of the data lying inside the ellipsoid centered on the cluster center.

iii. A mean-shift vector is calculated from the new and old cluster centers.

iv. This mean-shift vector is calculated iteratively until the cluster center does not change its position.

The parameters affecting the mean-shift segmentation are spatial kernel bandwidths, threshold for the minimum size of the region. Image detection, object tracking are the applications where mean-shift based segmentation is used.

Background histogram uses the difference in the deviation of image region with the created background model. This deviation is used to track the object in background subtraction method as shown in Figure 1.1.

Figure 1. 1 Background subtraction [4]

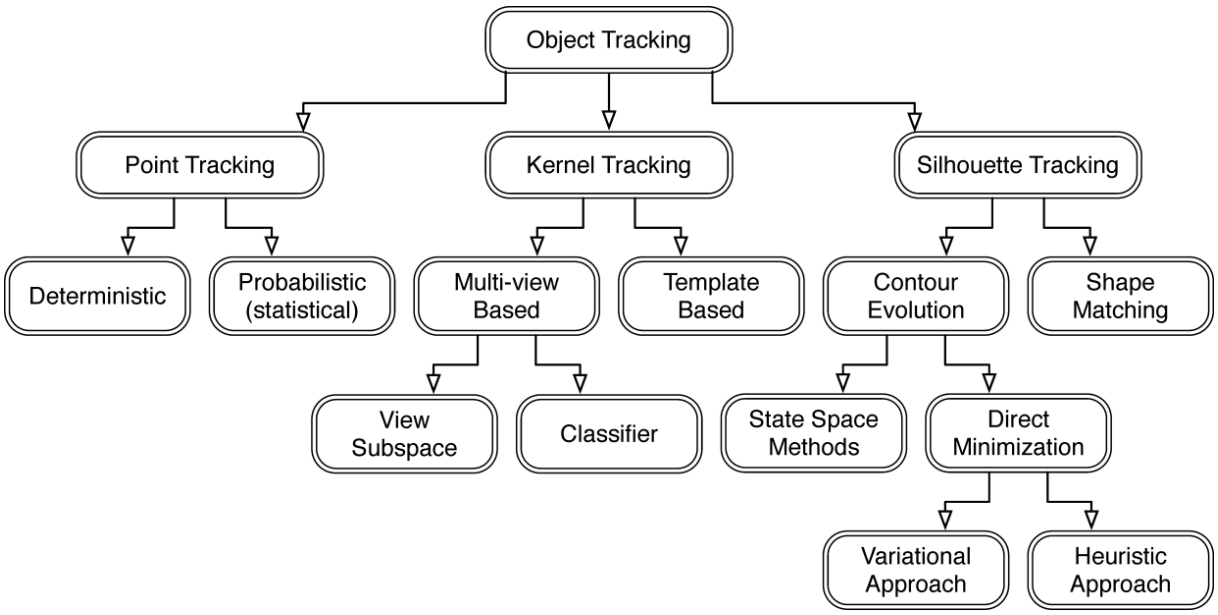

Figure 1. 2 Object tracking methodology [4]

1.5 Object Tracking

Object tracker generally estimates the trajectory of an object to be tracked over the time by detecting its position in each frame of the video. Several algorithms can be used to detect the region of the object and then tracker is used to track the object across frames. Figure 1.2 shows different methodology used in object tracking. Object tracking is classified into: (i) point tracking, (ii) kernel tracking, and (iii) silhouette tracking.

Point tracking includes, object characterized by points in each frame. These points are used to check the previous object state in a frame and are compared with the object in the subsequent frames. Point tracking is divided into two methods (i) deterministic method and (ii) Statistical method. MGE tracker and GOA tracker are deterministic methods of point tracking. Kalman filter and PMHT are statistical methods of point tracking algorithm.

In **kernel trackin**g kernel is refers to object's shape and its appearance. The kernel may have any shape such as rectangular or elliptical shape. Kernel is associated with a histogram. In this tracking method evaluation of the kernel motion is used to track object in each frame. Kernel tracking is classified into (i) template and density based appearance model (ii) Multi-view appearance model [4]. Mean-shift and KLT belongs to template based tracking, whereas eigen tracking and support vector machine are type of multi-view appearance models.

Silhouette tracking method is divided into (i) contour evolution (ii) matching shapes. State space models, variation methods and heuristics methods are part of contour evolution. Hausdorff, hough transform and histogram are matching shapes based tracking methods. Here information of the object region is retrieved from the density of appearance and shape models. Shape matching or contour evolution of object in each frame is done for tracking.

1.6 Literature Review

In the earlier years, tracking of point objects using infrared sensors in military applications was only done [2]. Later alpha-beta tracker paved the way for Kalman filter and extended Kalman filters which proved to be very useful for tracking. There are several literatures based on this and is reviewed here briefly. Tracking may be based on recognition [5], [6] or motion [7]. Tracking based on recognition is concerned in the recognition of object in successive images and extraction of its position. Its advantage is that it can be achieved in three dimensions and object translation and rotation can be estimated. But the problem here is that only recognized objects can be tracked, thus tracking performances are limited by high computational complexity.

Motion-based tracking rely on motion parameter estimation to detect the object. The advantage in this method is that, it can track any moving object irrespective of its size and shape. A recognition-based tracking system for traffic scenes is proposed by Koller [8]. This system is based on a prior knowledge about the shapes of vehicles and their motion to be tracked and also recognized in a scene. But it has high computational complexity and false matching combinations. Another system for tracking multiple vehicles in scenes of road traffic is given by Malik [9]. This system uses contour tracking algorithm where the position and motion of the contours are determined with the help of linear Kalman filter. But this system is limited only to shapes of a two dimensional object on the image plane. Again, tracking related to video surveillance in remote areas is proposed by Foresti [1]. Here statistical morphological skeleton is used for recognition and tracking which has low computational complexity and accuracy of localization is more.

Apart from various other algorithms, the mean-shift based algorithm for object tracking proposed by Comaniciu [10],[11] is a robust algorithm which uses color histogram that

solves the tracking problem due to scaling, rotation and partial occlusion. But the efficiency of this tracking algorithm is reduced when the target is not initialized properly or when there are more prominent background features. Further modified mean-shift algorithm proposed in [3] increases the efficiency of the mean-shift algorithm.

1.7 Problem Description

There are several challenges for a video tracker like foreground detection, illumination changes, occlusion, presence of clutter, robustness, accuracy, reducing computational complexity, etc. [12]. So while designing a tracker there are two major challenges which should be taken into account, i.e. either there may be similarity in appearance of the target and other objects in the background or there may be variation in appearance of the of the target itself [2]. When the features like shape or color obtained for the target is similar with the background, then it distracts the tracker and this phenomenon is known as clutter. Apart from this there may be change in appearance of the target like change in pose i.e. when the target moves, its appearance varies when projected into the image plane. Again appearance of the target also changes if the direction, light and color of the ambient light changes [2]. The performance of the tracker also degrades if the imaging sensor adds noise to the input image.

Another issue for the target to be lost from the scene is when the target is occluded by other objects in a particular scene. For example, a target moving behind a static object such as wall, table, etc. or other moving objects obstructing the view of the target [2]. Taking into account the above challenges, the problem defined for this research is for the development of a real-time object tracker that would work efficiently with maximum accuracy.

1.8 Objective

The key objective of this research is as follows:

- To develop an efficient VLSI architecture for an object tracking system which would track moving objects efficiently in a sequence of video frames.
- The video tracker should be able to estimate the trajectory of the target in a given video accurately.

- Along with this, when the algorithm is to be implemented on a hardware, the time for computation, hardware utilization, silicon area and power consumption for the overall system should be less.
- The algorithm to be used here for designing the tracker is the modified mean-shift algorithm [3].
- Initially the individual modules contained in the object tracking system is to be implemented on an FPGA platform and then design is to be done for the overall tracking system to interface with the outer world.
- As divider is the widely used arithmetic operation in image processing, so the detailed architecture of a divider has to be carried out with its functional verification.

1.9 Organization of thesis

The thesis organization is as follows:

Chapter-1: The first chapter gives an introduction to object tracking and various aspects of object tracking such as representation of an object, selection of features, detection of an object and methodology used for object tracking. Problem description along with the key objective of the thesis is also listed here.

Chapter-2: Out of different available object tracking algorithms, modified mean shift algorithm is chosen for implementation. This chapter presents MATLAB implementation of modified mean shift algorithm and also the simulation results are discussed.

Chapter-3: In this chapter hardware implementation of the modified mean shift object tracking is presented. The various module of a tracking system such as: kernel, density estimation, similarity estimation module, color transformation module and final mean shift tracking module are implemented by using hardware description language. The system functionality is verified by using simulation results.

Chapter-4: This chapter presents an architecture for another mean-shift algorithm and its HDL implementation. Finally the simulation results are discussed.

Chapter-5: Divider is a widely used mathematical operation in image processing applications. A possible architecture for a serial divider is discussed in this chapter and its simulation results are presented.

Chapter-6: Finally, a conclusion along with future work is drawn in this last chapter.

1.10 Conclusions

Object tracking is used in different applications of image processing and computer vision. A study has been done related to different issues of object tracking and the following are summarized:

- Objects are anything that is of interest for tracking and is represented by points, geometric shapes; object contour and skeletal model etc.
- The various features selected for object tracking are color, edge and texture. Color is being widely used feature for tracking but suffers from illumination changes.
- Detection of object can be done by point detectors, segmentation and background histogram.
- Different tracking methodologies such as point tracking, kernel tracking and silhouette tracking are used for tracking the object successfully.

CHAPTER 2

Mean-Shift Algorithm: MATLAB Simulations

- Introduction
- Mean-Shift Algorithm
- Modified Mean-Shift Algorithm for Tracking
- Algorithm for Modified Mean-Shift
- MATLAB Implementation of Modified mean shift algorithm
- Simulation Results in MATLAB
- Conclusions

2.1 Introduction

Object tracking is done to detect moving objects and follow the objects of interest by estimating the motion parameters [1], [2]. It plays a very crucial part in several computer vision applications [1], [2].These applications include those machines that can acquire, process and understand the input images and also react to the environment. Thus the goal of object tracking is to locate an object and find its orientation. The object to be tracked or the target may be of any kind. The object of interest to be defined depends on the specific application. The targets in building surveillance systems may be people whereas for some gaming applications the target may be faces or hands, again if its traffic control system then the targets may be vehicles. So reviewing several literatures and understanding the problems for tracking, the mean-shift algorithm is chosen and explained in the next sections.

2.2 Mean-Shift Algorithm

2.2.1 Introduction

Mean-shift algorithm for tracking moving objects was initially given by Comaniciu *et al.*[10]. If we have a set of samples, then according to this algorithm the modes or peaks in a density function is determined. This is a non-parametric method [12]. Its applications include segmentation, clustering, tracking, etc. Mean-shift based tracker tracks for a longer time and is more robust as compared to other trackers. This algorithm is basically an iterative process.

2.2.2 Properties of Mean-Shift

The properties of Mean-Shift algorithm are explained below:

- It is basically a tool for finding the modes i.e. peaks in a distribution or a set of data samples the ROI (region of interest).
- It has the direction same as that of gradient of the density estimate and its size also depends on the gradient.
- It is computed iteratively for obtaining the maximum density in the local neighborhood.
- Near maxima, steps are small and refined.

So for mean shift, we take a set of data points x_i , assign the weights g_i, sum them up, divide by the number of weights and then subtract initial estimate from it. Usually the highest mode is taken in a window [10],[11].

2.3 Modified Mean-Shift Algorithm for Tracking

Generally region of interest in a target includes background information, and if the content of this information is similar to the target, then accuracy of localizing the target decreases. To improve the target localization, Comaniciu *et al.* [3],[10],[13] proposed an algorithm called background-weighted histogram to represent the background features. This concept was useful for distinguishing the features of the target and target candidate region [3]. The mean-shift algorithm along with some modifications is explained below [3]:

The background is determined by the area surrounding the target and is represented as $\{\hat{o}_u\}_{u=1,\dots,m}$ with $\sum_{i=1}^{m} \hat{o}_u = 1$.The background is proposed to be three times greater than the size of the target. The coefficients responsible for transformation between the target model and candidate model is defined by [3],

$$\{ v_u = min\left(\frac{\hat{o}^*}{\hat{o}_u}, 1\right)\}_{u=1,..,m} \quad (1)$$

where $\hat{o}^*$ is the minimal non-zero value in $\{\hat{o}_u\}_{u=1,\dots,m}$. This decreases the weights of the background features having lower values of v_u. Let us consider the target model as

$$q_u = C \sum_{i=1}^{n} k(||x_i^*||^2)\, \delta[b(x_i^*) - u] \quad (2)$$

where , q_u is the probability of the u^{th} element of $\hat{q}$, δ is the delta function whose value is one when *u* has same intensity value at x_i^* , $b(x_i^*)$ associates pixel x_i^* to the histogram bin, *k(x)* is the isotropic kernel profile and *C* is a constant with $C = 1/\sum_{i=1}^{n} k(||x_i^*||^2)$.Now new target model is

$$q'_u = C' v_u \sum_{i=1}^{n} k(||x_i^*||^2)\, \delta[b(x_i^*) - u] \quad (3)$$

where $C' = \frac{1}{\sum_{1=1}^{n} k\left(\|x_i^*\|^2\right) \sum_{u=1}^{m} v_u \delta[b(x_i^*) - u]}$

Now moving to the next frame, the probability of bin *u* centered around *y* is given as

$$p_u(y) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)\delta[b(x_i)-u] \quad (4)$$

where $\hat{p}(y)$ is the target candidate model , $p_u(y)$ is the probability of u^{th} element of $\hat{p}(y)$, $\{x_i\}_{i=1,\ldots,n}$ are the pixels centered at *y* in the target candidate region, h is the bandwidth and C_h is the normalized constant with $C_h = 1/\sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$Now the new target candidate model is

$$p'_u(y) = C'_h \, v_u \sum_{i=1}^{n_h} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)\delta[b(x_i)-u] \quad (5)$$

where $C'_h = \frac{1}{\sum_{1=1}^{n} k\left(\left\|\frac{y-x_i}{h}\right\|^2\right)\sum_{u=1}^{m} v_u \delta[b(x_i^*)-u]}$ .The mean-shift iteration equation is given

as $y1 = \frac{\sum_{i=1}^{n_h} x_i w_i g\left(\left\|\frac{(y-x_i)}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{(y-x_i)}{h}\right\|^2\right)}$ (6)

For convenience here *g(x)*= -k'(x) and taking $g_i = g\left(\left\|\frac{y-x_i}{h}\right\|^2\right)$ we get

$$y1 = \frac{\sum_{i=1}^{n_h} x_i w_i g_i}{\sum_{i=1}^{n_h} w_i g} \quad (7)$$

The transformations(3) and (5) stated above reduce the background features but ultimately gives the same result as original mean-shift algorithm and thus an advanced mean-shift algorithm is stated [3] here. Considering u′ as the bin index corresponding to x_i, where x_i is the point in the candidate and assuming $\delta[b(x_i) - u'] = 1$, the weight , w'_i is computed using background-weighted histogram technique as

$$w'_i = \sum_{u=1}^{m} \sqrt{\frac{q'_{u'}}{p'_{u'}(y)}} \quad (8)$$

But for corrected background weighted histogram technique the transformation is done only for the target model and not the candidate model and hence the new weight formula is given as

$$w''_i = \sum_{u=1}^{m} \sqrt{\frac{q'_{u'}}{p_{u'}(y)}} \quad (9)$$

Now substituting (3),(5) and the normalization constants value in (9) we get

$$w''_i = \sqrt{\frac{C'}{C}} \times \sqrt{v_{u'}} w_i \quad (10)$$

Neglecting the constant terms, we get $w''_i = \sqrt{v_{u'}} w_i$ (11)

From (11) we infer that if the feature at point *i* is in the background region is prominent, then the corresponding value of $v_{u'}$ is small and thus the weight is also decreased. This further increase the convergence speed of the mean-shift algorithm and the target is not lost from the selected region in any of the frames. Also the background model must be dynamically updated for robust tracking because there may be occlusion or illumination changes. So in order compute this, initially the background features $\{\hat{o}'_u\}_{u=1,\ldots,m}$ and $\{v'_u\}_{u=1,\ldots,m}$ in the current frame is calculated. Then the Bhattacharya similarity between the old background model $\{\hat{o}_u\}_{u=1,\ldots,m}$ and the new background model $\{\hat{o}'_u\}_{u=1,\ldots,m}$ is computed as [3].

$$\rho = \sum_{u=1}^{m} \sqrt{\hat{o}_u \hat{o}'_u} \quad (12)$$

If ρ is smaller than a specified threshold, then it signifies changes in the background and then $\{\hat{o}_u\}_{u=1,\ldots,m}$ is updated by $\{\hat{o}'_u\}_{u=1,\ldots,m}$ and also $\{v'_u\}_{u=1,\ldots,m}$ is updated as $\{v_u\}_{u=1,\ldots,m}$. The transformed target model q'_u is then calculated using (3). If ρ is greater than the threshold, then there is no need of any update in the background model.

2.4 Algorithm for Modified Mean-Shift

1. Initially the following parameters are computed:
    - Target model q_u using equation (2).
    - Background-weighted histogram $\{\hat{o}_u\}_{u=1,\ldots,m}$and then $\{v_u\}_{u=1,\ldots,m}$using equation (1).
2. Then the transformed target model q'_u is calculated using equation (3).
3. The position y_0 of the target candidate region in the previous frame is initialized.
4. Let the iteration value k=0 initially.

5. Target candidate model $p_u(y_0)$ in the current frame is calculated using equation (4).
6. Weights w''_i is calculated using equation (11).
7. The new position y_1 of the target candidate region is calculated using (7).
8. Let $\|y_1 - y_0\| = d$ and y_1 is updated to y_0 , and k=k+1
9. The following parameters are set
    - Mean-shift threshold ε_1 to a default value (0.1).
    - Maximum iteration number, N.
    - Background model update threshold ε_2 to a default value (0.5).
10. If $d < \varepsilon_1$ or k $\geq$ N then calculate $\{\hat{o}'_u\}_{u=1,\ldots,m}$ and $\{v'_u\}_{u=1,\ldots,m}$ based on the tracking result of the current frame. If ρ (12) is smaller than ε_2 , then $\{\hat{o}_u\}_{u=1,\ldots,m}$ is updated as $\{\hat{o}'_u\}_{u=1,\ldots,m}$ $\{v'_u\}_{u=1,\ldots,m}$ is updated as $\{v_u\}_{u=1,\ldots,m}$ and q'_u is updated by equation (3). Iteration is stopped and moved to step 4 for next frame. Otherwise step 5 is executed.

2.5 MATLAB Implementation of Mean Shift Algorithm

MATLAB implementation of the mean shift algorithm is done to verify and understand the how mean-shift algorithm is used for tracking objects successfully. A set of standard code available at the MathWorks [14] site by Sylvain Bernhardt is used for this purpose. The flowchart of the mean shift tracking module is explained with a flowchart as shown in Figure 2.1. Stating about the mean-shift tracking algorithm [10],[11], if we are having sufficient amount of data samples, then its key objective is to determine the densest region of the given distribution. This is a non-parametric method to find modes in a probability density function. Mean-shift algorithm is based on determination of the mean-shift vector and the iteration continues till it converges [10]. The algorithm is well explained with the help of a flow chart in Figure. 2.1. Initially for localization of the target object, a region of interest or window is selected in the current frame [1], [3], [10]. At this point kernel estimation is done. A required feature space is then selected in order to represent the target in the current frame called as target model. In this case the color histogram of the target or density function is determined to represent the target model [3].

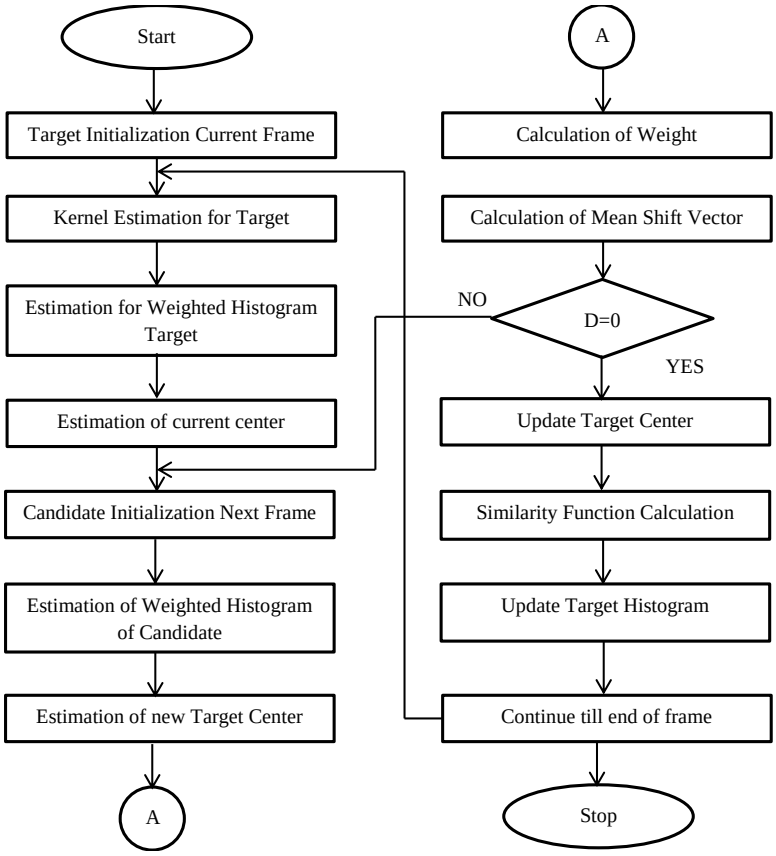

Figure 2. 1 Flow chart of mean-shift algorithm

Then moving to the next frame, the target called as candidate model is determined. Now if the target is same in this frame, then it should have the same probability density function or histogram as the target in the previous frame. This can be determined by computing the similarity measure between the target model and the candidate model. Then the displacement vector is calculated and the process continues till the mean-shift vector converges to a very small value.

Hence the main components of the MATLAB code are as follows:

1. Density estimation.
2. Parzen window and gradient estimation.
3. Similarity function.
4. Mean shift tracking.

Density estimation is the process of estimating the density of the image. The image of interest is represented by color histograms with a kernel profile of k. A patch is drawn in the image with a height H and width W . A column one dimensional array q is used to save estimated density for a patch of T. Two density estimations p and q are evaluated with a kernel profile of k.

Parzen window calculates the mask and its gradient by taking different types of kernel along with X and Y axis. The different types of kernel are Uniform, Triangular, Epanechnikov, Gaussian.

Similarity function estimates the similarity between the above estimated two density estimations p and q. q is the density of reference patch and p is the density estimation of the candidate one.

Mean-shift tracking module is used to implement the mean-shift algorithm and find out the tracking values of a selected image. A movie is imported and an image is selected for tracking. Variables like start index, similarity threshhold and numbers of maximum iterations to have convergence and kernel types are declared. Parzen kernel window is calculated. The image RGB colors of the image are converted to index colors so that color probability function can be computed. Then similarity between the tracking in first frame to last frame is evaluated.

2.6 Simulation Results in MATLAB

A standard video sequence of 52 frames is used here for verifying the tracking algorithm. The kernel type used here is the Epanechnikov kernel. The simulation results done in MATLAB for the ping-pang ball sequence is shown below. The original mean-shift algorithm shown in Figure 2.1 is compared with the modified mean-shift algorithm using corrected background-weighted histogram shown in Figure 2.2 and Figure 2.3 for frames numbered 1, 10,26,38,45 and 52. The rectangle in blue color shows the target initialization.

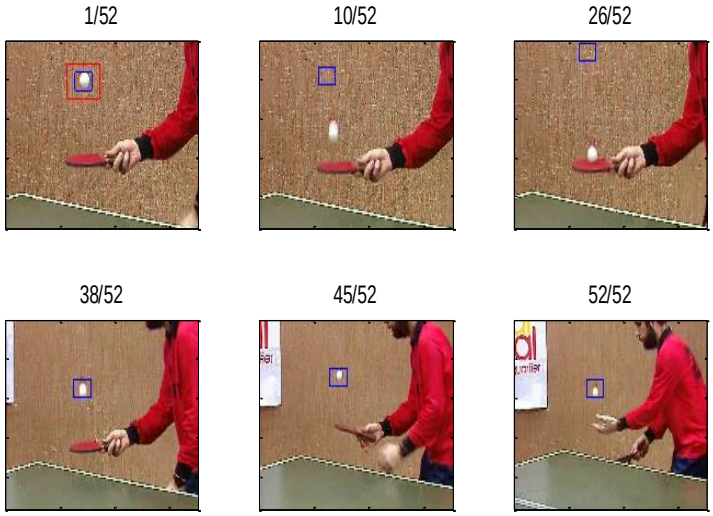

Figure 2. 2 Using original mean-shift algorithm

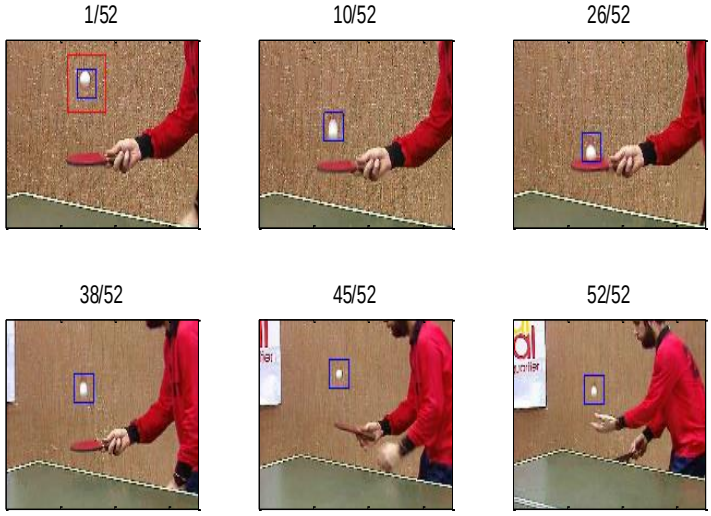

Figure 2. 3 Using modified mean-shift algorithm

2.7 Conclusions

It is seen that the modified mean-shift algorithm tracks the moving object in this case ball as shown in Figure 2.3 more accurately than the original mean-shift algorithm as shown in Figure 2.2. The modified algorithm has the following advantages:

- Requires less computation time, more reliable.
- Histogram of the background is also determined which helps to separate target and background features.

A VLSI architecture designed and verified for original mean-shift algorithm is explained in chapter 3.

CHAPTER 3

VLSI Architecture for Object Tracking System-I

- Introduction
- VLSI architecture for Object tracking system-I
- Kernel Estimation Module
- Density Estimation Module
- Similarity Co-efficient Estimation Module
- Mean-Shift Tracking Module
- Color Space Transformation module
- Device Utilisation for FPGA Implementation
- Conclusions

3.1 Introduction

Implementation of an efficient video processing algorithm on an FPGA platform is emerging with several challenges which need to be addressed by the design engineers. An algorithm with less computational complexity and hardware utilization for real-time applications is explained in chapter 2. The VLSI architecture for tracking the required object according to the mentioned algorithm is presented in this chapter. The tracking algorithm based on mean-shift iteration technique tracks accurately the target object in a sequence of video frames in presence of any occlusion or variation in the appearance. This chapter describes the implementation of an object-tracking algorithm on FPGA. The programming language used to configure FPGA is VHDL. The modules are implemented on FPGA target device XILINX xc5vlx110t-2ff1136.

3.2 VLSI architecture for Object tracking system-I

The block diagram for the tracking system is shown in Figure 3.1.

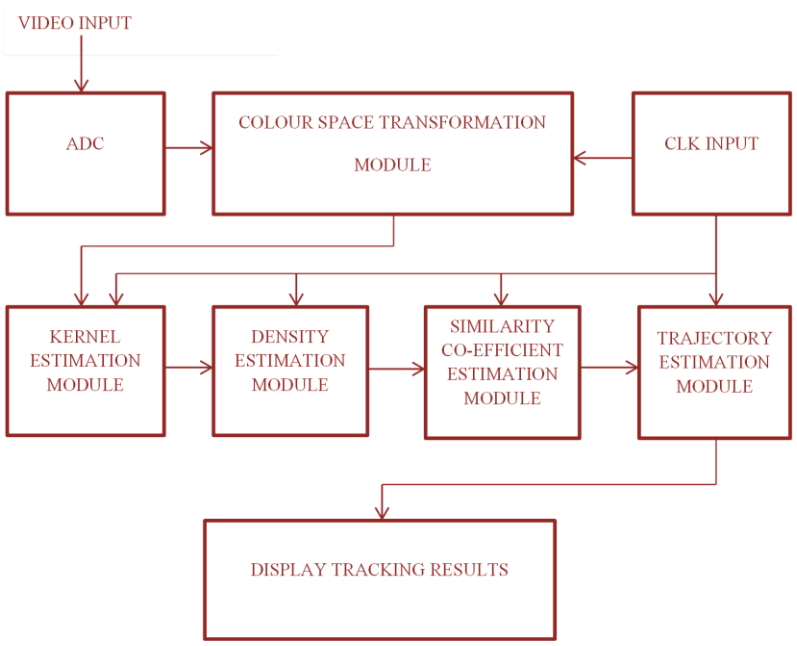

Figure 3. 1 Block diagram for object tracking system

The different modules implemented using VHDL are the color transformation module, kernel estimation module, density estimation module, determination of similarity coefficient and the

mean-shift tracking (trajectory estimation) module whose sequence of operation is shown in the Figure 3.1. Each module is designed by creating a finite state machine (FSM) which provides the most efficient way of creating hardware using hardware description language (HDL).

VLSI architecture for the MATLAB code available at the MathWorks [14] site by Sylvain Bernhardt, which is implemented using VHDL is given in Figure 3.2. The process starts with determination of the kernel matrix by taking height and width of the required image as input, then the density estimation (DET and DEC) where the color space transformation is not included here. But a separate module is designed and verified for color space transformation which is explained in details in section 3.7. So after the density estimation, similarity coefficient (SIM-FUN) is determined and then the mean-shift tracking (MS-TRACKING) process is carried out.

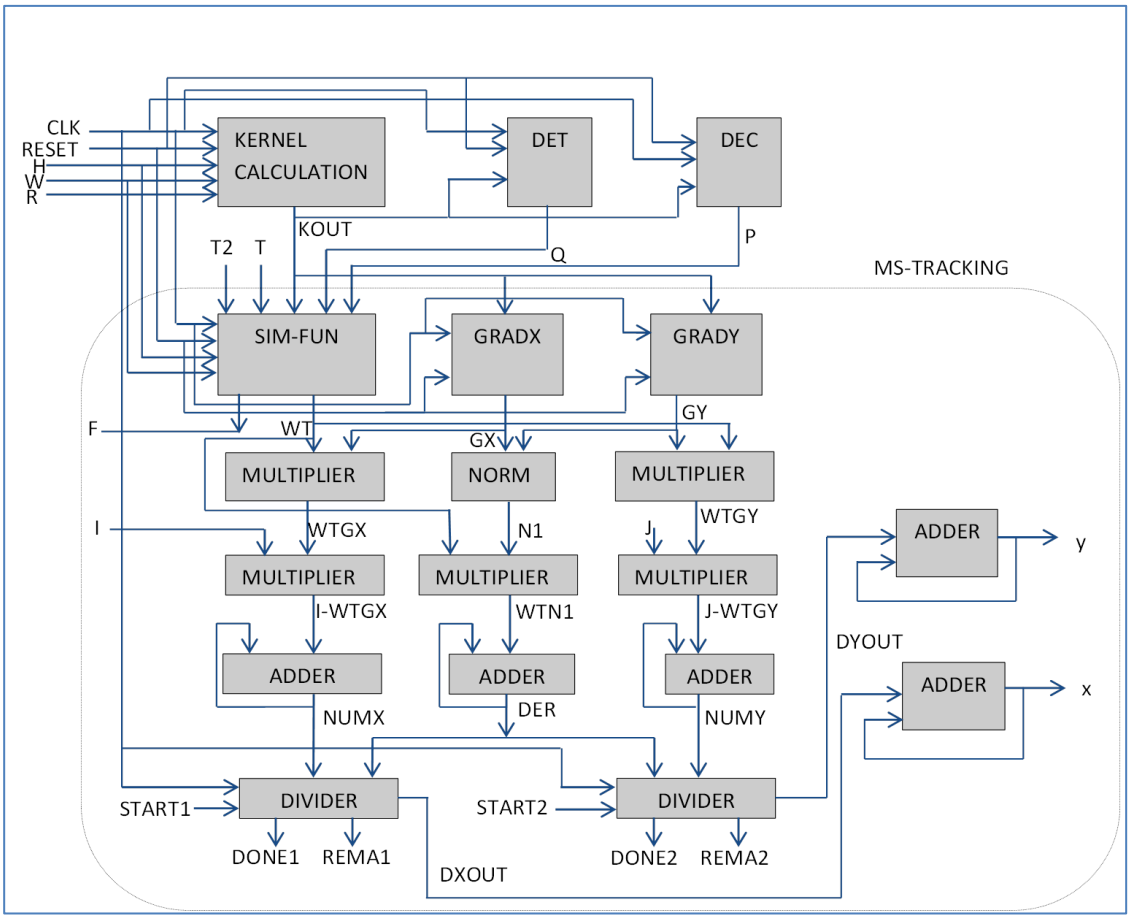

Figure 3. 2 VLSI architecture for object tracking system-I

Here in the figure blocks GRADX and GRADY is used to determine the gradient values of the kernel matrix in x and y direction respectively. All these blocks are explained in detail in the below given sections.

3.3 Kernel Estimation Module

The kernel estimation is also known as Parzen Window technique. Here the mask or kernel is determined. Kernel defines the magnitude of the weight to be assigned to the input pixels of the target image. It depends on the size of the input target image. Here height is denoted as H, width as W and radius as R of the window is given as the input and the array of values for the kernel(k) is determined at the output. Here i and j are the loops which execute till they attain the values of H and W respectively. The kernel used here is the Epanechnikov kernel. The kernel architecture is shown in Figure 3.3.

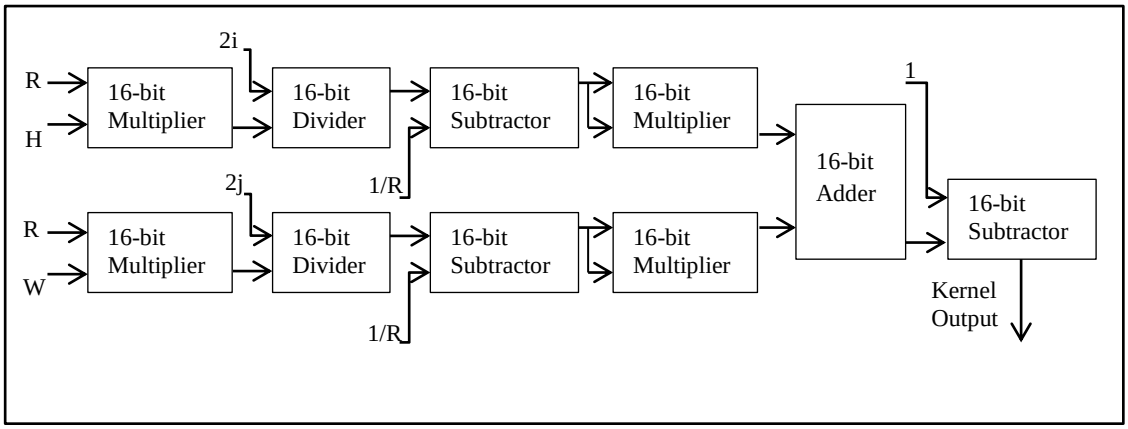

Figure 3. 3 Kernel estimation architecture

3.3.1 VHDL Simulation and Verification with MATLAB Results

The simulation results obtained for the kernel matrix in VHDL were verified to be correct for all the values of i and j when compared with MATLAB results. The results obtained for H=W=16 and R=1 for all values of i and j obtained in MATLAB are shown in Table 3.1.

i , j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0.000	0.000	0.000	0.000	0.094	0.172	0.219	0.234	0.219	0.172	0.094	0.000	0.000	0.000	0.000	0.000
2	0.000	0.000	0.047	0.188	0.297	0.375	0.422	0.438	0.422	0.375	0.297	0.188	0.047	0.000	0.000	0.000
3	0.000	0.047	0.219	0.359	0.469	0.547	0.594	0.609	0.594	0.547	0.469	0.359	0.219	0.047	0.000	0.000
4	0.000	0.188	0.359	0.500	0.609	0.688	0.734	0.750	0.734	0.688	0.609	0.500	0.359	0.188	0.000	0.000
5	0.094	0.297	0.469	0.609	0.719	0.797	0.844	0.859	0.844	0.797	0.719	0.609	0.469	0.297	0.094	0.000
6	0.172	0.375	0.547	0.688	0.797	0.875	0.922	0.938	0.922	0.875	0.797	0.688	0.547	0.375	0.172	0.000
7	0.219	0.422	0.594	0.734	0.844	0.922	0.969	0.984	0.969	0.922	0.844	0.734	0.594	0.422	0.219	0.000
8	0.234	0.438	0.609	0.750	0.859	0.938	0.984	1.000	0.984	0.938	0.859	0.750	0.609	0.438	0.234	0.000
9	0.219	0.422	0.594	0.734	0.844	0.922	0.969	0.984	0.969	0.922	0.844	0.734	0.594	0.422	0.219	0.000
10	0.172	0.375	0.547	0.688	0.797	0.875	0.922	0.938	0.922	0.875	0.797	0.688	0.547	0.375	0.172	0.000
11	0.094	0.297	0.469	0.609	0.719	0.797	0.844	0.859	0.844	0.797	0.719	0.609	0.469	0.297	0.094	0.000
12	0.000	0.188	0.359	0.500	0.609	0.688	0.734	0.750	0.734	0.688	0.609	0.500	0.359	0.188	0.000	0.000
13	0.000	0.047	0.219	0.359	0.469	0.547	0.594	0.609	0.594	0.547	0.469	0.359	0.219	0.047	0.000	0.000
14	0.000	0.000	0.047	0.188	0.297	0.375	0.422	0.438	0.422	0.375	0.297	0.188	0.047	0.000	0.000	0.000
15	0.000	0.000	0.000	0.000	0.094	0.172	0.219	0.234	0.219	0.172	0.094	0.000	0.000	0.000	0.000	0.000
16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 3. 1 MATLAB results for kernel matrix

The VHDL output (y) is shown in 8-bit binary format where the most significant bit represents the integer value and the remaining bits represent the fractional part. The VHDL simulation results are shown in Figure 3.4. In the waveform we can see that for i=1 and for all values of j from 1 to 4 the value of y is zero, for i=1 and j=5 the binary value obtained in VHDL is 00001100 which represents 0.09375 and for i=1 and j=6 the binary value is 00010110 which represents 0.1718 which is same as the value obtained in MATLAB shown in Figure 3.4 and so on.

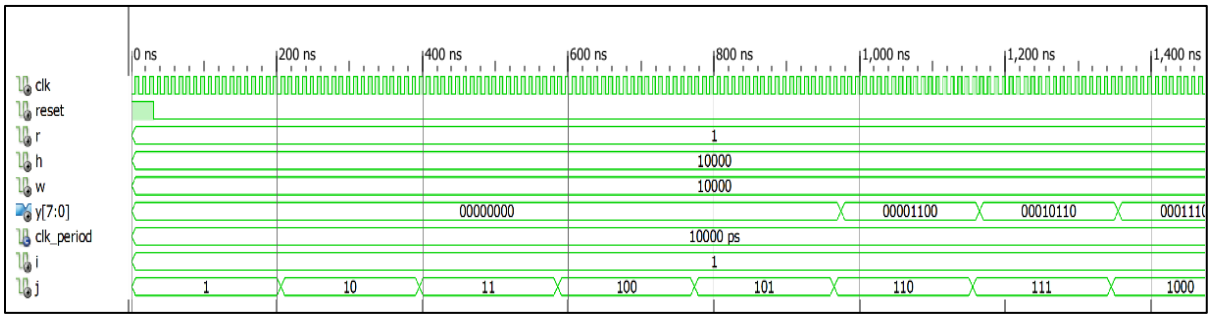

Figure 3. 4 VHDL simulation results for kernel module

3.4 Density Estimation Module

For the density function of the data samples, specifically the color histogram is computed. The input for this module is the kernel profile (k) obtained from the kernel estimation

module, the target initialization matrix (T) of size same as height (denoted as H) and the width (denoted as W) of the kernel matrix. Again i and j are the loops iterating till they attain the values of H & W. The architecture is shown in Figure 3.5. Here q (T) represents the density function of the target model with index equal to the individual elements of T. The output (D) obtained is the density of the samples after normalization i.e. after dividing q (T) with sum of all values present in the kernel matrix. In Figure 3.2, DET block is used for determining the probability density for target and DEC block is used for determining the probability density for candidate.

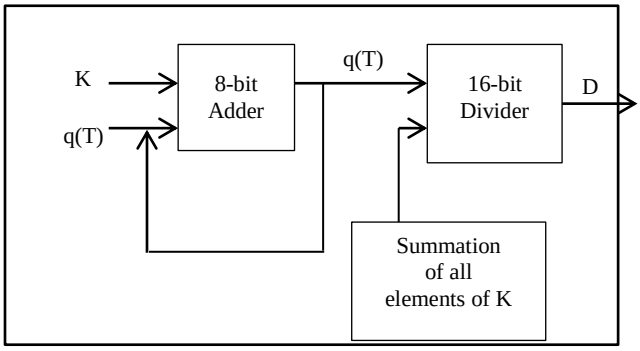

Figure 3. 5 Density estimation architecture

3.4.1 VHDL Simulation and Verification with MATLAB Results

For the density estimation in VHDL the value of H and W is taken as 16 and the matrices T and K are also initialized with certain values to verify. Values for the target model are stored in an array of size 4096 and each value is stored as 8-bit binary form where all the bits represent a fractional number. For example binary value 00000010 between 16,000 ns and 16,100 ns in the waveform window represents 0.0078, 00000110 represents 0.0234 and so on. These values are verified with MATLAB results. The waveforms after i=15 and j=15 are shown in Figure 3.6.

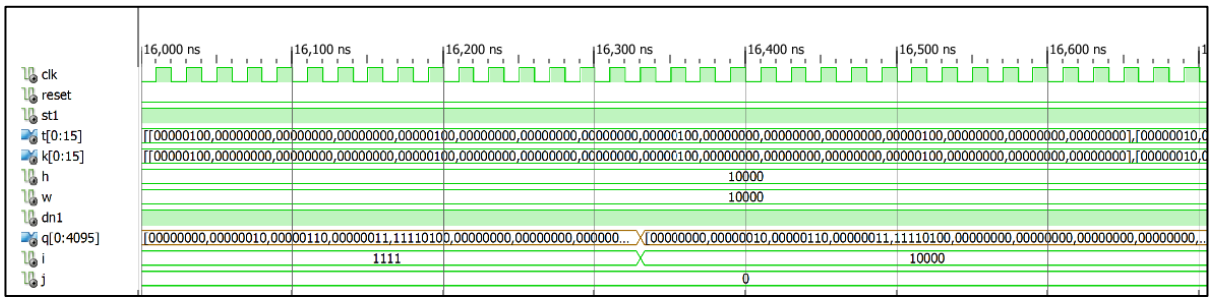

Figure 3. 6 VHDL simulation results for density estimation module

3.5 Similarity Co-efficient Estimation Module

The density function for target model (q) and candidate model (p) can be obtained as stated in section 3.4 and 3.3. Now to compare these two density functions, we need to compute the similarity function, popularly known as Bhattacharya Coefficient. This includes a distance metric such that, minimizing the distance corresponds to maximizing the similarity function. This indicates higher similarity between the two density functions. The input for this module are the density estimations of target model (q), candidate model (p), kernel estimation(k), the candidate initialization matrix(T2) of size same as height(H) and the width(W) of the window. Also *i* and *j* are the loops same as mentioned earlier. The architecture is shown in Figure 3.7. Here q(T2) and p(T2) represents the density function of the target model and candidate model respectively with index equal to the individual elements in T2 matrix.

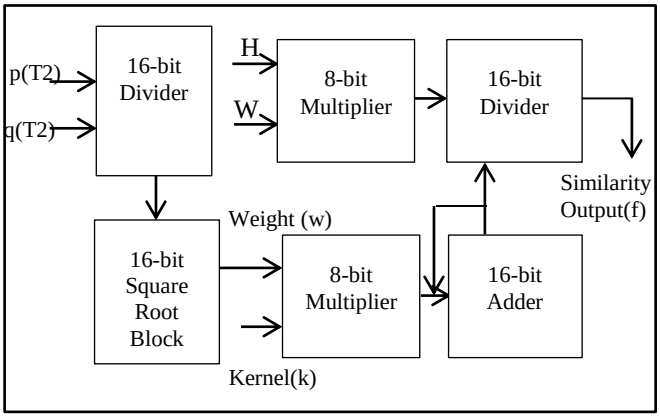

Figure 3. 7 Similarity co-efficient estimation architecture

3.5.1 VHDL Simulation and Verification with MATLAB Results

For the similarity measure in VHDL the value of H and W is taken as 16 and the matrices T2 and K are also initialized with certain values to verify. The weight matrix (w) obtained in MATLAB is shown in Table 3.2. In VHDL simulation results shown in Figure 3.8, w is of 8 bits where the first four bits represent the integer part and the remaining four represent fractional value of a number. For example the binary value obtained for 1.2247 is 00010011 which is equivalent to 1.18 and the binary value for 1 in MATLAB is 00010000.

1	1	1	1	1	1	1	1	1.2247	1.2247	1.2247	1.2247	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	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	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	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	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	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
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	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	1	1	1	1

Table 3. 2 MATLAB results for weight matrix

The similarity measure (f) is of 16 bits where the first 8 bits represent integer value and the remaining bits represent fractional value. Here f obtained for MATLAB is 0.0156 and for VHDL 0000000000111100 which represents 0.234375.

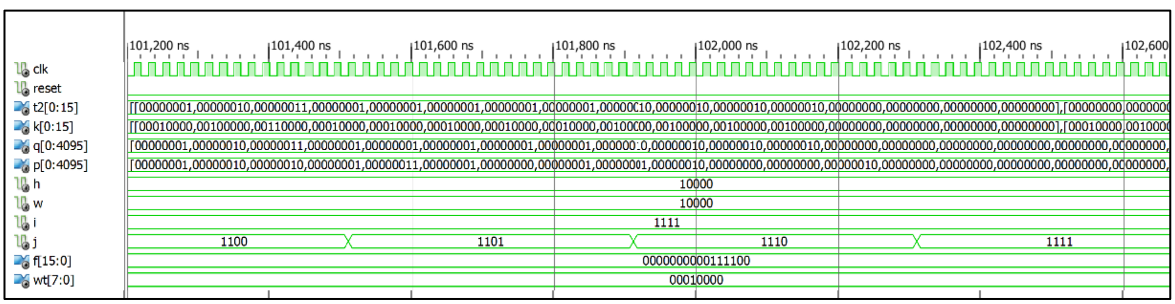

Figure 3. 8 VHDL simulation results for similarity co-efficient estimation module

3.6 Mean-Shift Tracking Module

Mean shift tracking module includes the similarity function estimation module mentioned in section 3.5, modules for determining gradient in x and y direction i.e. GRADX and GRADY as shown in Figure 3.2 a block for determining the norm (NORM) of a number used for finding the norm of mean-shift vector and other arithmetic blocks. This module uses the mean-shift algorithm and finds out the tracking values of a selected image. Initially the similarity between the tracking in first frame to last frame is evaluated. Thus this algorithm is used to converge the x and y location of the object in an image from first frame to next

frame. Here mean-shift vector is determined and target location is updated. The architecture for this module according to the algorithm is included as a sub-block in the complete architecture shown in Figure 3.2. The significance of GRADX, GRADY and NORM are explained in next sub-sections.

3.6.1 Gradient Estimation Module

Gradient in image processing is defined as the change in the intensity or may be color of the image in a particular direction. In our architecture it is used to find the gradient of the kernel matrix values in x and y direction. The algorithm used for this block is given in the equation below [14],

$$A_i = \frac{1}{2} \sum_{i=2}^{N-1} A_{i+1} - A_{i-1} \qquad (1)$$

A_i refers to a row element in a matrix. Gradient function calculates the central difference between data points and the gradient at the end points, where i=1 and i=N (row indices), is calculated by finding the difference between the end point value and the next adjacent value within the row. In this way the gradient in x- direction can be calculated. Similarly using the above equation and substituting j for i (column indices), the gradient in y-direction is determined.

3.6.1.1 VHDL Simulation

The VHDL simulation results for gradient in x and y direction is shown in Figure 3.9 and 3.10 respectively.

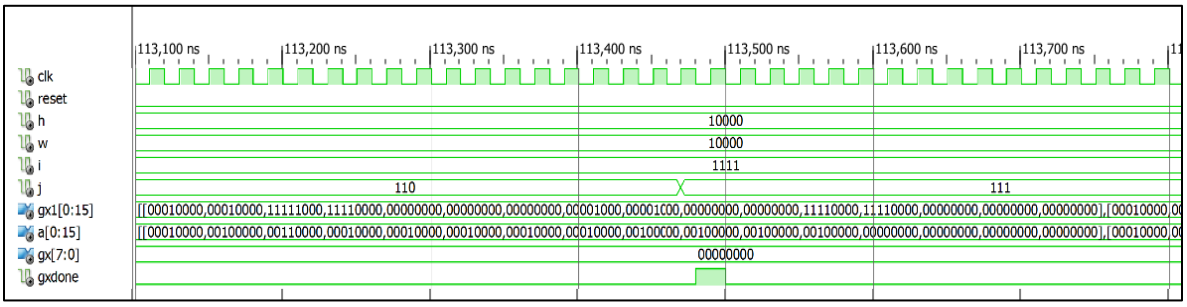

Figure 3. 9 VHDL simulation results for gradient of a matrix in x-direction

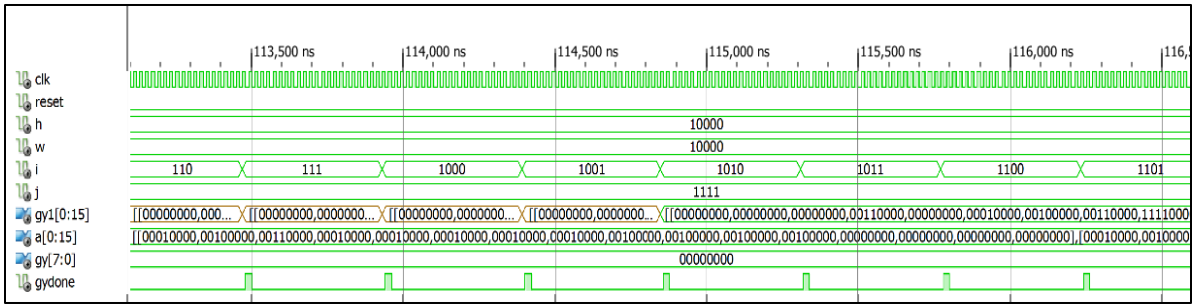

Figure 3. 10 VHDL simulation results for gradient of a matrix in y-direction

Here ‘a’ shown in the waveform is a 16x16 matrix whose gradient in x-direction is stored in gx1 (16x16 matrix) and gradient in y-direction is stored in gy1 (16x16 matrix). The values obtained for gx1 and gy1 are of 8 bits where the first four bits represent integer value and the remaining bits represent the fractional value.

3.6.2 Norm Estimation Module

Norm is used to assign a definite positive size to a vector. The algorithm used here for determining the norm of a vector is the sum of the square of the gradient values gx and gy as given in the equation below,

$$Norm=\sqrt{(gx^2 + gy^2)} \qquad (2)$$

3.6.2.1 VHDL Simulation

The VHDL simulation results for norm of gradient values gx and gy in x and y direction respectively is shown in Figure 3.11. Here gx and gy are o 8 bits where the first four bits represent the integer value and the remaining four bits represent the fractional part of a number. As shown below gx has value 1111100 which represent -0.5 and gy has value 00000000 which represents 0. Thus output n is 00001000 that represent 0.5 which is verified correct.

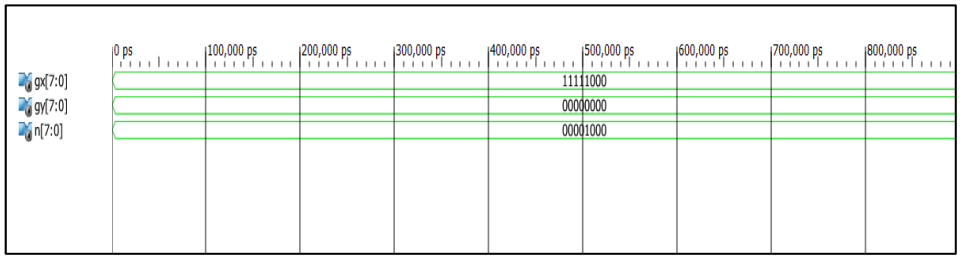

Figure 3. 11 VHDL simulation results for norm of gradient values in x and y direction

3.6.3 VHDL Simulation and Verification with MATLAB Results

Thus the final tracking is done by integrating the values from other blocks. The VHDL simulation results are shown in Figure 3.12. The verification with MATLAB results is shown in Table 3.3 The output numx and numy are of 32 bits where first 16 bits combine to represent an integer and the remaining bits represent fractional value.

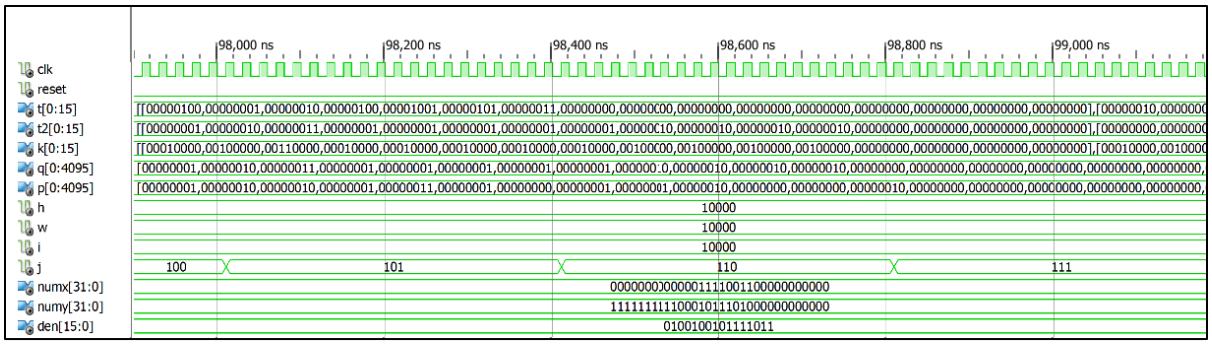

Figure 3. 12 VHDL simulation results for mean-shift tracking

	MATLAB	VHDL
numx	7.6124	7.5937
numy	-6.1566	-5.8125
den	67.2573	73.4804
dx	0.1132	0.1015
dy	-0.09	-1.2929

Table 3. 3 Verification with MATLAB results

3.7 Color Space Transformation module

This module plays a crucial role in all video and image processing systems. There are several color spaces RGB, YUV and HSV. So depending on the application the required feature space is used. We can convert from one color space to other. Generally a little change in intensity of any color class results in error in object location. The color of an image is determined by the combination of intensity values of these three color components. These components in an image are generally R, G, B values and we need to transform it to a

suitable color space. Thus color transformation is a process where the representation of a color in one coordinate is converted to another. Transforming the components is just scaling the intensity values of the components in a given color space. The architecture of color transformation module is as shown Figure 3.13.

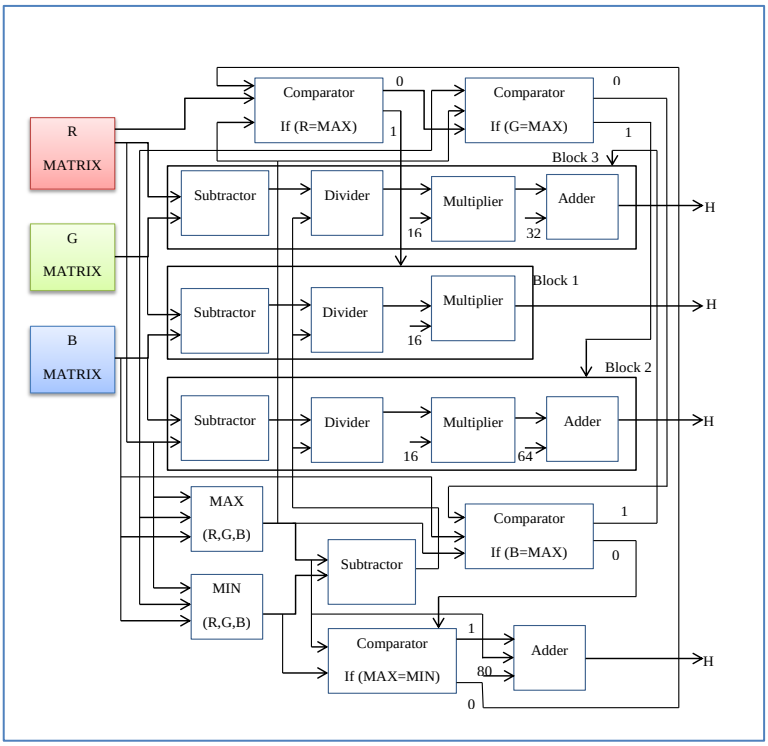

Figure 3. 13 Architecture of color space transformation module

Figure 3.13 shows the architecture of color transformation module. A pipelined architecture is designed which helps in simultaneous operation of different blocks [13]. Initially the R, G, B values of an image are imported from MATLAB and then stored in a memory as shown in the Figure 3.13 as R-matrix, G-matrix and B-matrix. Then the next operation is to find the maximum (max) and minimum (min) value among these R, G and B values. The max and min value of individual matrix is first calculated and then again maximum of the three max values is determined and similarly the minimum of the three min values is determined. The min value is subtracted from max so that the result (max-min) obtained can be used for normalization as the limited range for max is 255 and min is 0. Then simultaneous subtraction of the individual matrices with one another is done, i.e. B is subtracted from G (G-B), R is subtracted from B (B-R) and G is subtracted from R (R-G). The max value is

compared with the min value with the help of a comparator. If the result is true or logic ’1’, then the max value is added with 80 and the hue component H is obtained .But if the result is false or logic ‘0’, then R is compared with the max value. Here all the elements of R matrix are compared with the max value and if any of the elements is same as max, then result is true and block1 is operated.

Block1 comprises of a divider and a multiplier where the G-B value is divided by max-min and then multiplied by 16 to get the H component. If the result for the comparator is false, then all the elements of G matrix are compared with the max value and if max value matches with any of the matrix elements in G, then block2 is operated. Block2 has same elements where B-R is divided by max-min and then multiplied by 16 and then an adder is used to add the result with 32 to obtain the H component. Now if the comparator result is false, then B matrix elements are compared with the max value and if the result is true, then block3 operates which is exactly same as block 2 only the division occurs between R-G and max-min. So the process continues till complete iteration.

3.7.1 VHDL Simulation

For VHDL implementation the elements of the R, G and B matrices are represented in 8-bit where first four bits represent the integer value and the remaining four bits represent the fractional value.

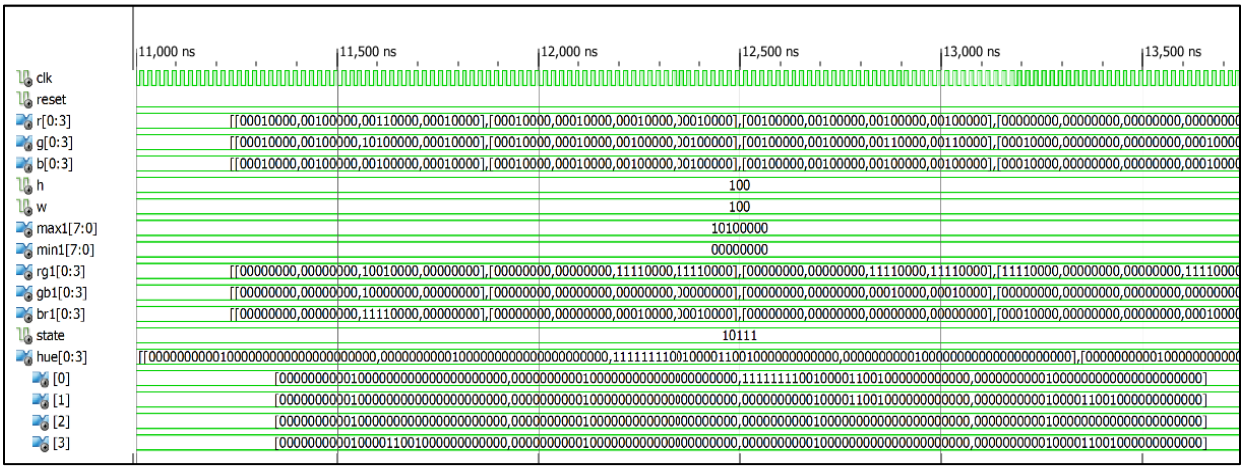

Figure 3. 14 VHDL simulation results for color space transformation

The hue component obtained is in 32-bit where its first 16 bits represent integer value and the remaining 16 bits represent the fractional value. The VHDL simulation results are shown in

Figure 3.14. We can observe that, R, G and B values are input here and the max value (max1) in simulation window comes from G matrix and the hue value obtained is verified with MATLAB results. Here H (1,1) is 32 and the VHDL output is 00000000001000000000000000000000 and so on.

3.8 Device Utilization for FPGA Implementation

The utilization of hardware resources and the minimum period (Tmin) estimated for all the above stated modules are given in Table 3.4. Hardware utilization depends on the logic and architecture style which is used for designing. Since the input is image here care should also be taken for less utilization of memory. The input image matrix used for all the modules is 16x16 and also the height and width is taken as 16. . The target device used here is xc5vlx110t-2ff1136.

Modules	Number of Slice registers		Number of Slice LUTs		Number of fully used LUT-FF pairs		Minimum Time Period (Tmin) in ns
	Used	Utilization (%)	Used	Utilization (%)	Used	Utilization (%)	
Kernel	134	0	327	0	99	27	7.122
Similarity	174	0	3029	4	153	5	15.173
GRADX	48	0	1171	1	47	4	5.310
GRADY	48	0	1171	1	47	4	5.310
Norm	92	0	0	0	24	3	NA
Mean-shift Tracking	4494	6	6512	9	294	2	18.644

Table 3. 4 Device utilization for FPGA implementation

3.9 Conclusions

All the modules are implemented successfully in VHDL and verified with the MATLAB results. The VLSI architecture described for each module describes the simplicity of the algorithm used. It is seen that the logic used for designing a particular architecture is very simple even if the size of the image is increased. Here the size of the image matrix is taken to be 16 x 16 and it is found that the hardware utilized is very less in percentage. This feature is advantageous when we integrate all the modules to obtain the complete object tracking system. But one limitation of the algorithm used here is that, for determining the density models the elements of the kernel matrix are used as index values for the output array. This hampers the accuracy of the tracking result. Also if the color transformation is carried out, then negative values may be obtained which becomes an invalid index value for the density array. This problem can be solved by using a slight change in the algorithm used here. The new algorithm along with the VLSI architecture is explained in detail in chapter 4.

CHAPTER-4

VLSI Architecture for Object Tracking System-II

- Introduction
- Tracking algorithm
- VLSI architecture for modified tracking algorithm
- VHDL Simulation
- Conclusion

4.1 Introduction

The VLSI architecture for mean-shift based tracking system as described in the previous chapter was based on the determination of kernel or window of a given image matrix. The probability density function of the target and the candidate was then calculated where the kernel matrix served to be the input of the density estimation module. Even the similarity between the two models were determined whose value was again dependent on the values in the kernel matrix. The said architecture was also dependent on the color space transformation of the input image matrix leading to more utilization of hardware resources. In this chapter a modified VLSI architecture is presented which is independent of the kernel matrix values. Here the window for the target is determined by the co-ordinates of the center and half size of the window. Instead of performing the color space transformation, a suitable technique is used to determine the R (red), G (green) and B (blue) values of the input image matrix and use it for certain parameters which would directly calculate the probability density function of the target and candidate models. The detailed algorithm and architecture are explained in the next sections.

4.2 Tracking Algorithm

The algorithm here is based on the MATLAB code given by the authors in [3]. Mean-shift iteration technique is used but here certain alterations are made to the first algorithm as stated in chapter 3. The algorithm is shown in terms of a flow chart in Figure 4.1. Initially the values for constant input parameters like minimum convergence threshold, maximum iteration number and increase size of the window is set. Once the frames of an input video for an object to be tracked are captured, then they are processed one after another in a sequence. The frame arriving first is considered to be the current frame and the target model is determined from this frame. When the second frame comes the candidate model is determined.

Starting with the first frame, the center value is initialized and then the position of the target window is determined as given in the following equations 1-4 [3],

rmin = center1- whalfsize1 (1)

rmax= center1+ whalfsize1 (2)

cmin= center2- whalfsize2 (3)

cmax= center2+ whalfsize2 (4)

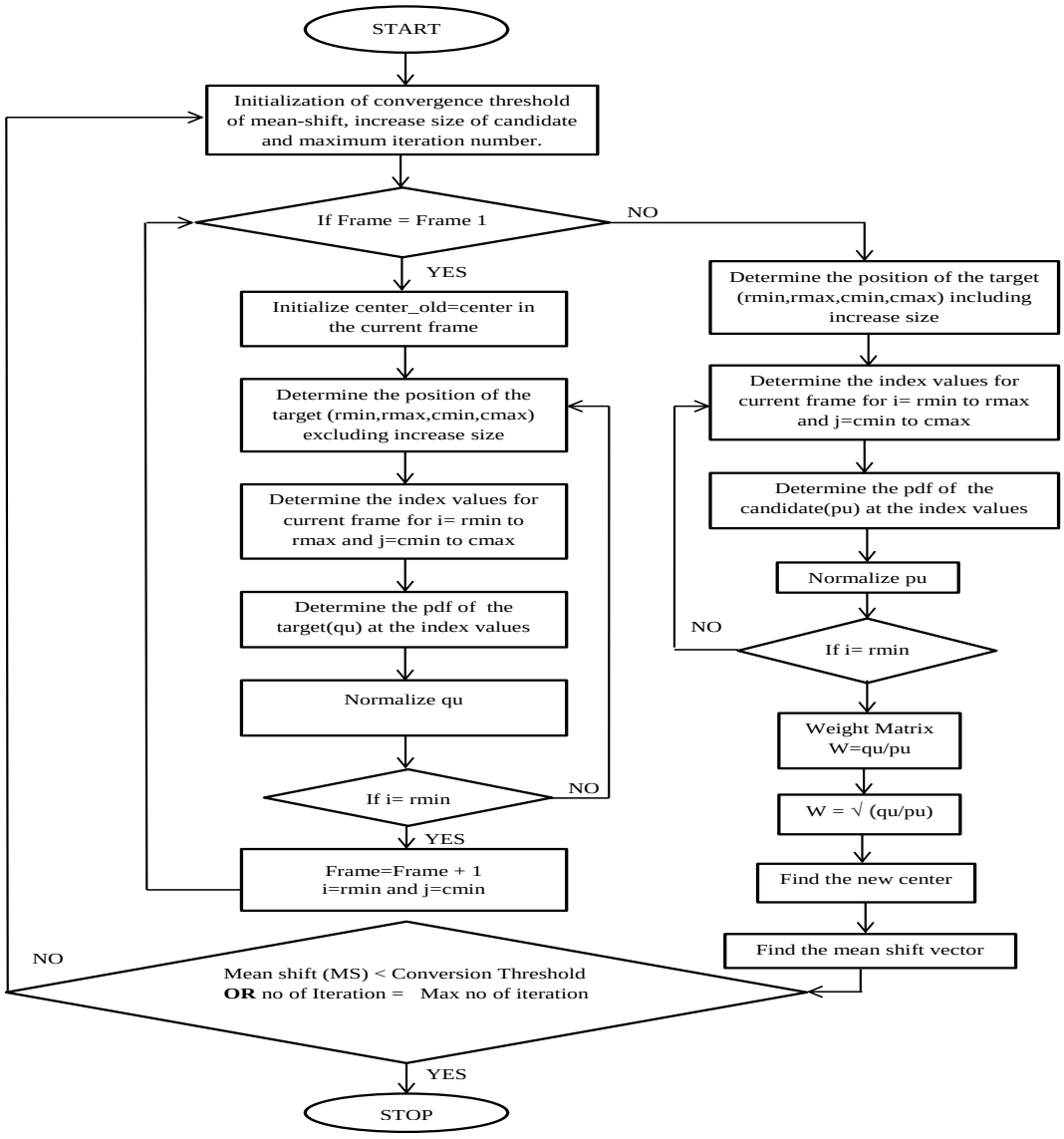

Figure 4. 1 Flow chart for modified tracking algorithm

In the above equations, center1 and center2 are the initial position of the target in x and y direction respectively. Also whalfsize1 and whalfsize2 are the half size of the window in x and y directions respectively. Here rmin, rmax, cmin and cmax also determine the range for

which the loops have to iterate. The iterating loop i, ranges from rmin to rmax and j ranges from cmin to cmax. For all these values of i and j, R, G and B values are found. Using these values, index is determined and target model (p) is calculated for the calculated index values and then normalized. When i value reaches rmax, the second frame is evaluated. The position of the candidate window can be determined by the following equations 5-8,

rmin = center1- whalfsize1-incre (5)

rmax= center1+ whalfsize1+incre (6)

cmin= center2- whalfsize2 - incre (7)

cmax= center2+ whalfsize2 +incre (8)

Here incre is the increase size of the window. The candidate model (q) is then determined in the same process. Now in this second frame height and width of the input image is determined and also the weight matrix of size (rmax × cmax) is found as per equation 9 given below,

$$w=\sqrt{(q/p)} \quad (9)$$

The values of i and j are stored in arrays x1 and x2 of size 0 to (rmax × cmax) respectively. The new centroid is then found both in x (center_new1) and y (center_new2) direction according to equation 10-11 as follows,

center_new1 = (x1*w)/ (summation of all elements in ‘w’ matrix) (10)

center_new2 = (x2*w)/ (summation of all elements in ‘w’ matrix) (11)

The mean-shift vector is determined as given in equation 12,

MS= √((center_new1-center1)^2+(center_new2-center2)^2) (12)

Now if the mean-shift vector is less than the convergence threshold or the number of iterations is greater than the maximum iteration number, then the iteration converges and the tracking is said to be completed. The VLSI architecture for this algorithm is explained in the next section.

4.3 VLSI Architecture for Modified Tracking Algorithm

The architecture for the said algorithm is designed and shown in Figure 4.2. The input to the tracking system comprises of center1, center2, whalfsize1 (WHS1), whalfsize2 (WHS2), height and width of the input image along with the clock and the reset signals. Here size of the input matrix is taken to be as16x16. So, two sets of three input matrices of size 16x16 are taken. One set of matrices named as Rmat1, Gmat1 and Bmat1 as shown in the Figure 4.2 is for determining the values of target model and another set of matrices named as Rmat2, Gmat2 and Bmat2 as shown in the Figure 4.2 is for determining the values of candidate model.

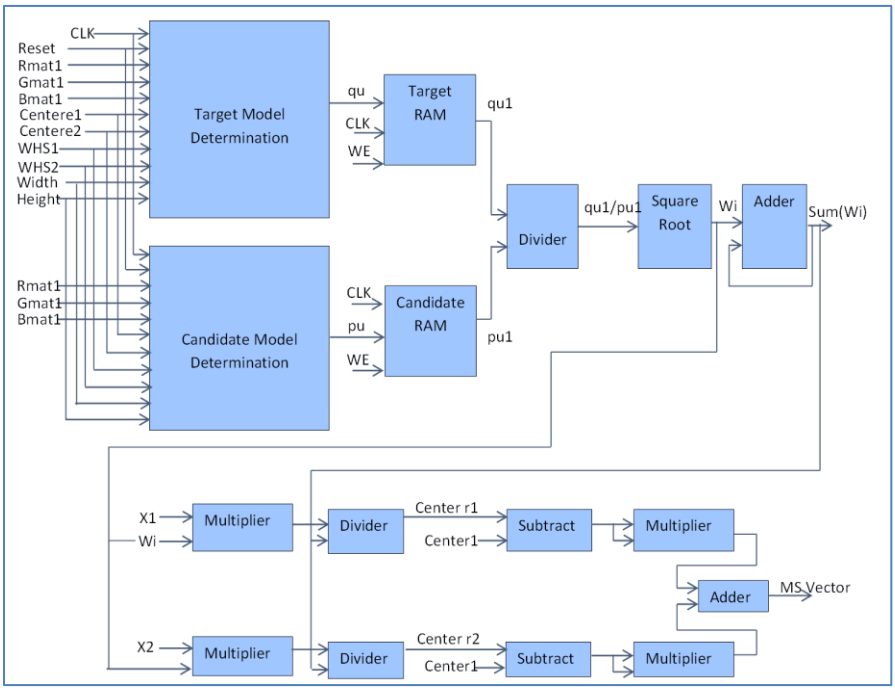

Figure 4. 2 Complete VLSI architecture-II for object tracking

Once the first frame is obtained, the index values are determined as per the architecture shown in Figure 4.3 and stored in the index RAM shown in Figure 4.2. The target model is then determined according to the architecture shown in Figure 4.4 and stored in the target RAM as per the index values when its enable signal (WE) is high. Now when second frame is arrived the new index values are calculated and stored in the index RAM using the same architecture as Figure 4.3 and the candidate model is determined according to the same

architecture as Figure 4.4 but with different input values and stored in the candidate RAM when its enable signal (WE) is high.

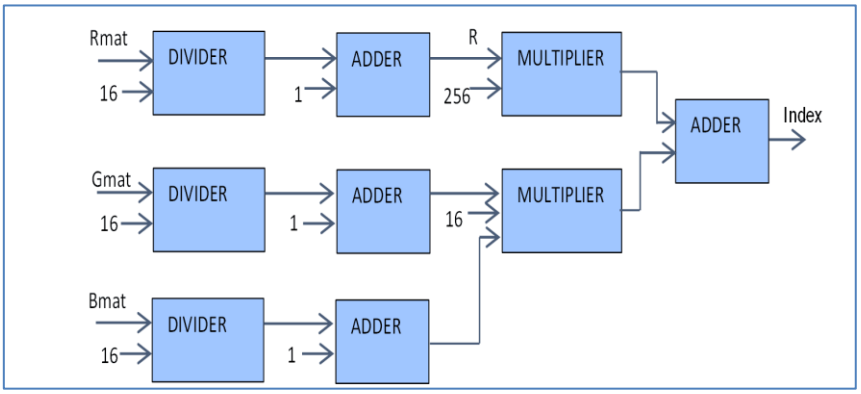

Figure 4. 3 Architecture for determining the index values

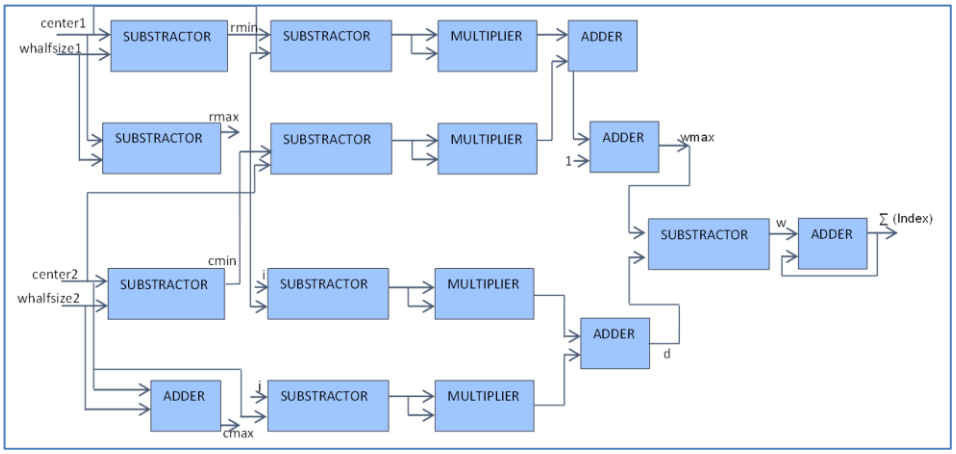

Figure 4. 4 Architecture for determining the target model/candidate model

In order to find the weight matrix, the values of ‘q’ and ‘p’ for all iteration values need to be retrieved from the RAM. So the enable signal for both the target RAM and candidate RAM is made low for read operation. At this step all the values of ‘q’ and ‘p’ for index values determined as in second frame is retrieved. Now weight matrix can be determined as per equation (9) and the next steps i.e. determination of weight matrix (wi), new center (center r1 and center r2) and mean –shift vector (MS vector) is found according to the algorithm has the architecture as shown in Figure 4.2 .The main blocks of the algorithm is determined with the help of various arithmetic blocks of which the divider and square root block plays a crucial role.

4.4 VHDL Simulation

The VHDL simulation is carried out in the FPGA target device XILINX xc5vlx110t-2ff1136. The results for target model, candidate model, weight matrix and new center is shown in Figure 4.5, 4.6, 4.7 and 4.8 respectively. Figure 4.5 shows that when frame (in the simulation waveform the variable assigned for frame is frame1)is zero i.e. when first frame is arrived, at index values 2458, 2730, 2731, etc. different values of qu (target model) are obtained. Here qu value is in 16-bit binary format where all the bits combine to form a fractional number. For example 1111000000000000 represents 0.9375 and so on which is verified with the MATLAB results.

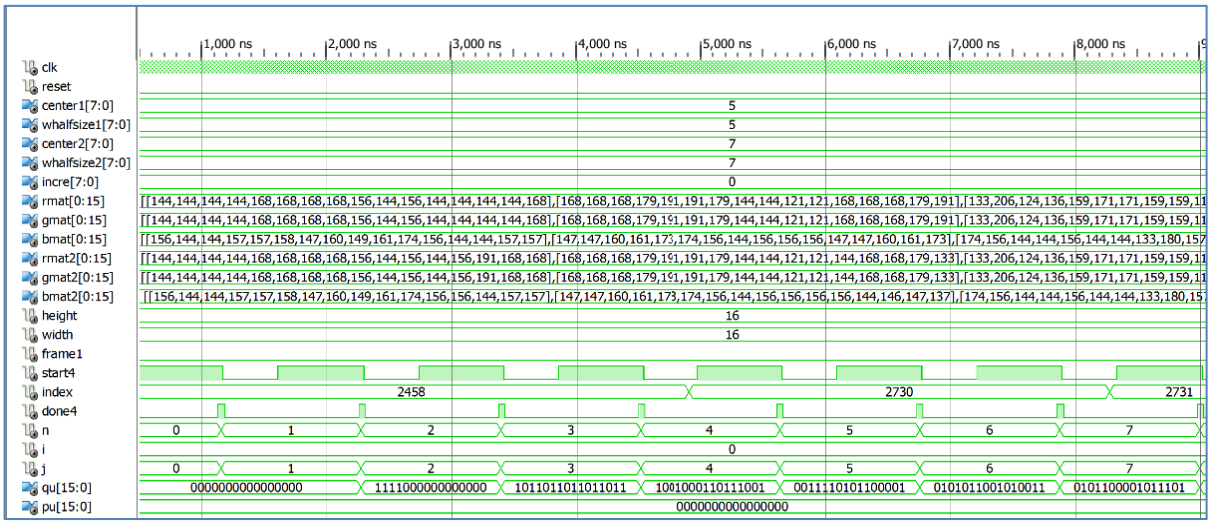

Figure 4. 5 VHDL simulation showing the values for the target model

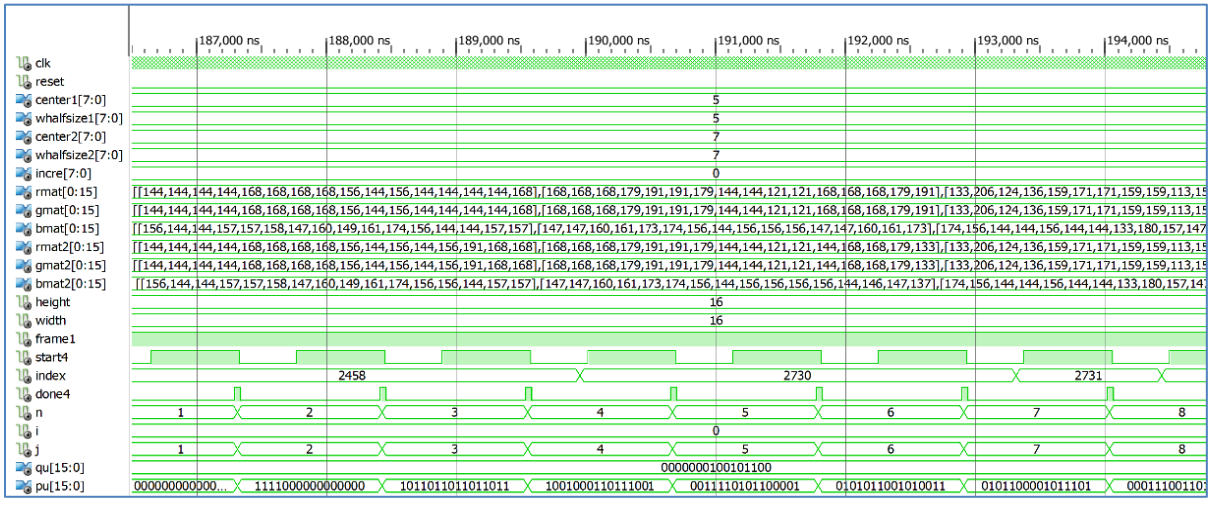

Figure 4. 6 VHDL simulation showing the values for the candidate model

Similarly Figure 4.6 shows that when frame is one i.e. when second frame is arrived, at index values 2458, 2730, 2731, etc. different values of pu (candidate model) are obtained. Here pu value is in 16-bit binary format where all the bits combine to form a fractional number.

Figure 4.7 shows the values obtained for weight matrix. Here the variable assigned is wi_sqt, which is of 8-bit where the first four bits represent the integer part of a number and the remaining last four bits represent the fractional part. For example 00010000 represents value '1', 00001111 represents 0.9375 and so on. These values are verified successfully with MATLAB results.

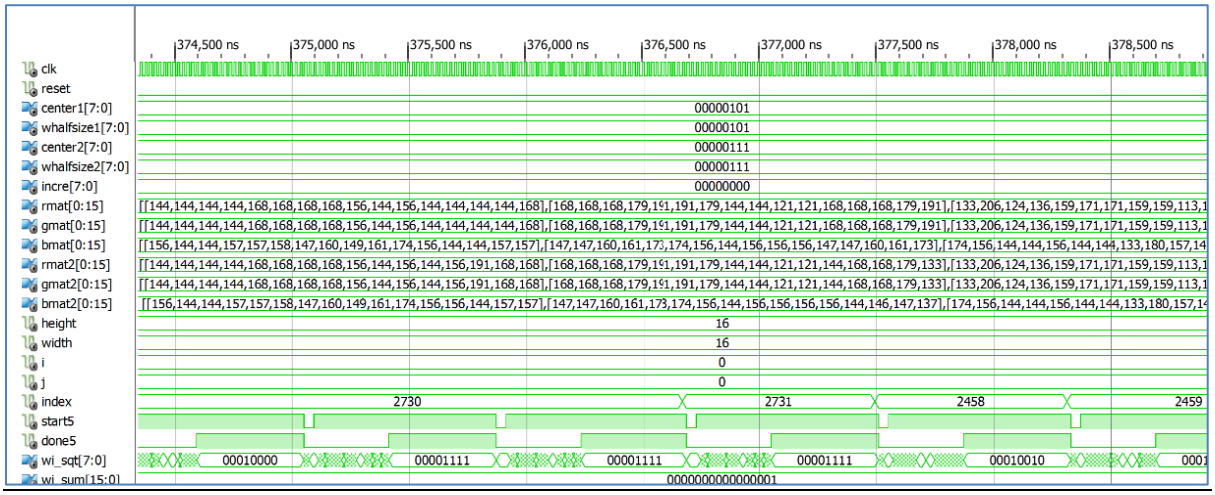

Figure 4. 7 VHDL simulation showing the values for weight matrix

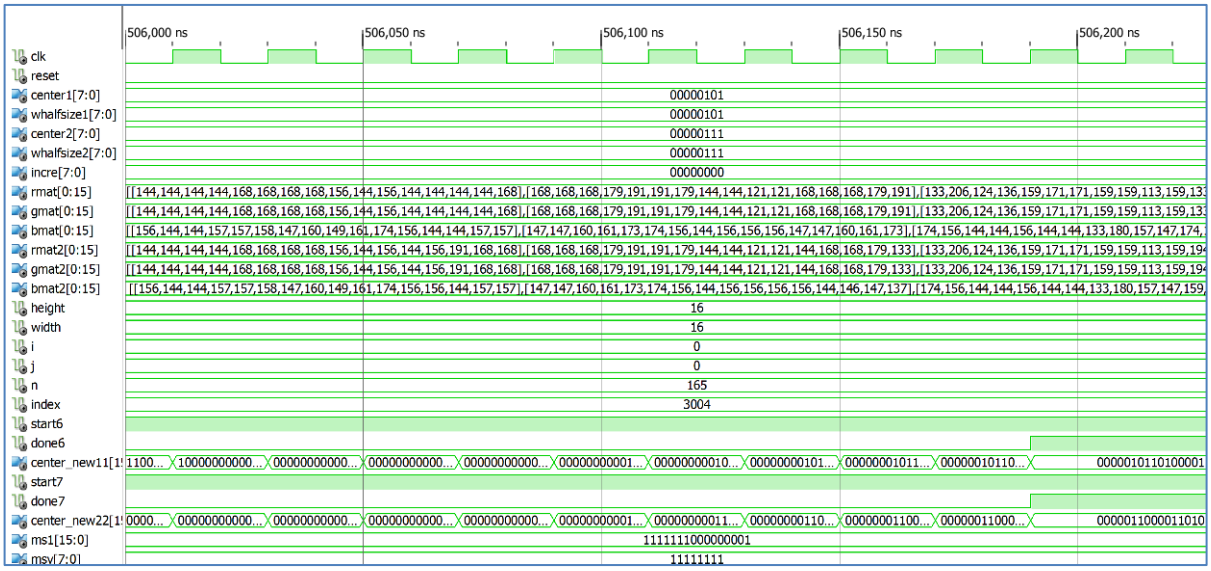

Figure 4. 8 VHDL simulation showing the values for new center and mean-shift vector

Figure 4.8 shows the values obtained for new center i.e. center_new11 and center_new22 in x and y direction respectively. These variables are of 16-bit in binary representation where the first eight bits from left represent the integer part and the remaining eight bits represent the fractional part of a number. Variable center_new11 has value 0000010110100001 i.e. 5.6892 when done6 is one or high. Variable center_new22 has value 0000011000011010 i.e. 6.1015 when done7 is one or high.

4.5 Conclusions

Hence the complete architecture for the modified algorithm is implemented successfully in VHDL. The new center values obtained are 5.6892 (approx. 6) and 6.1015 (approx. 6). The process continues to find the new centroid in every iteration till the the mean-shift vector converges to zero i.e. when the value of mean-shift vector is less than the mean-shift convergence threshold or till the iteration value reaches the maximum iteration number. After the final iteration, the new center moves along the entire sequence of video frames and thus signifying that the object in a video is being tracked to much accuracy. In terms of optimisation of this system in terms of VLSI architectures, we can go for optimizing the divider and the square root blocks present in the architecture as shown in Figure 4.2, 4.3 and 4.4. A digital serial divider is discussed in the next chapter showing certain techniques to implement it at transistor level.

CHAPTER 5

VLSI Architecture for Serial Divider

- Introduction
- Basic Division Schemes and Algorithms
- Serial Division Algorithm
- Architecture of Serial Divider Block
- Implementation of Serial Divider Architecture
- Simulation Results Serial Divider
- Conclusions

5.1 Introduction

In modern processor based system design, a system is made up of processor which is the heart of the system. The processor is accountable for performing computational functionality, sharing of data with peripherals, reading from and writing into memories and finally overall functioning of the system. The major time consumption of a processor involves the memory related data transfer. Apart from this, the processor also consumes time to perform intensive computational operations. Design of such computational blocks always involves a variety of issues, such as efficient area optimization and reducing the timing of computation. These two major issues become more relevant when the processor based system is targeting to applications such as digital signal processing, digital image and video processing. In such applications, the computational operations involved are addition, subtraction, multiplication, accumulation and division. In image and video processing where an image is represented by a 256x256 matrix, the mathematical computation of such a huge matrix involves added time as well as massive area of the computational blocks.

There are two ways of performing these computations in a processor based system. One way is by using software based approach, where this approach being sequential in nature suffers from performance flaws. The other approach involves performing the computations by using dedicated hardware blocks. The parallelism property of hardware block helps greatly in speeding up of the computational ability of the system thus increasing the overall performance of the system [18]. Hence to achieve a high level of performance, the massive mathematical computations are carried out using dedicated hardware block.

A survey of a group of people working in the area of processor architecture and implementation reveals that, division is the most time consuming operation in comparison to addition, subtraction and multiplication [18]. Division is the most complex and important arithmetic operation whose performance can be enhanced by using dedicated hardware divider blocks.

Intel Pentium 4 processor uses SRT division (Sweeney[19], Robertson[20] and Tocher[21]) module for performing division operation. In 1994 Intel had a loss of $475 million, due to an error in the SRT block of the floating point unit of Pentium processor [18].

This necessitates more investigations and research in the area of divider algorithms and their efficient architecture implementations which accelerate the modern processor based system design.

5.2 Basic Division Schemes and Algorithms

Division is a complex arithmetic operation which is used widely in all mathematical computation. Hence designers have to put efforts to design divider block by taking different existing algorithms. Most of the algorithms, implemented the divider block by using adders and multipliers as building blocks [22].

The basic division algorithm is classified into two categories: 1. Subtract and Shift 2. Multiplicative. The first method is the usual paper and pencil method which is same as the division of a decimal number by another decimal number. In case of the decimal division, each xi of dividend X and yi of divisor Y are selected from binary number {0,1} and the quotient is selected from radix r, which is equal to be 2^K (i.e., 2,4,8 and so on).This method with radix r iterates the recurrence step of subtract and shift. Hence it is also termed as digit recurrence algorithms [23].

The digit recurrence algorithm calculates one digit of the quotient in each iteration. Here the shifted remainder is important for the calculation of quotient bit. The comparison of shifted partial remainder with the multiples of divisor is used to calculate the quotient bit. Digit recurrence method offers many advantages. Implementation of this algorithm is simple. It uses controlled adder/subtractor cells. The accuracy level of this algorithm is more but suffers from timing complexity as compared to multiplicative based division scheme [24]. There are three types of digit recurrence algorithm i.e. restoring, non-restoring and SRT. Radix-2 restoring division algorithm uses a quotient digit q_i from a set of digit {0, 1}.The quotient digit is '1' when $R_i' = 2R_{i-1}-Y \geq 0$, else it is '0', where Y is divisor and R_i' is the partial remainder. If $Q_i=0$ then Y is again added with R_i'. Hence this method is also known as

the restoring method of division. This method is iterated for a specific n-bit of accuracy level [23].

Non-restoring division algorithm can be distinguished from restoring one as the divisor is not added back again with the partial remainder. Hence it is called as non-restoring algorithm.

Radix-2 SRT algorithm, chooses the quotient bit Q_i from a digit set {-1,0,1}. Q_i is selected as -1 or 0 or 1 based on $2R_{i-1} < 0.5$ or $-0.5 \leq 2R_{i-1} < 0.5$ or $0.5 \leq 2R_{i-1}$. When quotient is '0' no addition or subtraction operation is performed. This method is introduced by Sweeney, Robertson and Tocher, hence it is called as SRT division algorithm.

Multiplicative division is also called as functional iteration algorithm. Functional iteration techniques are used to converge from an initial estimation towards the quotient with required precision. Though it is faster than other digit recurrence algorithm, but it suffers from complexities in steps and more computations are required to get the final remainder. Hence the overall complexity of this algorithm is more [25]. This methodology of division is used in commercial applications such as mainframe computers and some microprocessor [23]. There are two commonly used functional iteration algorithm: 1. Newton-Raphson convergence equation 2. Taylor –Maclaurian expansion (Gold Schmidt's algorithm).

In another algorithm, the reciprocal of divisor Y is calculated using Newton-Raphson method. The approximation begins with the reciprocal of the divisor, U_0. Then 1/Y can be determined by an iterative calculation of $U_{i+1}=U_i . (2-U_i.Y)$. The quotient is then found by multiplying the dividend X with 1/Y [25].

Gold Schimdt's algorithm uses a method where the fractional value remains same when the numerator and the denominator are multiplied by the same number. For example, $X/Y = (X. D_0. D_1. D_2. \ldots)/(Y. D_0. D_1. D_2. \ldots.)$ where D_i's are selected so that $Y. D_0. D_1. D_2\ldots$ approaches to '1' and $X. D_0. D_1. D_2\ldots$ approaches to X/Y. The quotient can be obtained through iterative calculation of $D_i = 2-Y_i$, $Y_i+1= Y_i. D_i$ and $X_i+1 =X_i$ [25].

5.3 Serial Division Algorithm

Many division algorithms have been proposed in various literatures. The most common algorithm is digit recurrence algorithm which is further classified into restoring, non-restoring and SRT algorithm. Digit recurrence also uses add/subtract-shift in iterative process to perform a division. Non-restoring is the simplest digit recurrence algorithm which can be used to design a digital serial divider. Digital serial approach for division is simple and cheap in comparison to digital parallel division approach, which is bit expensive and complex. Digital serial division approach is comparatively slower than parallel approach but the complexity of hardware design and cost puts a trade off in selecting the serial approach over the digital parallel approach [24].

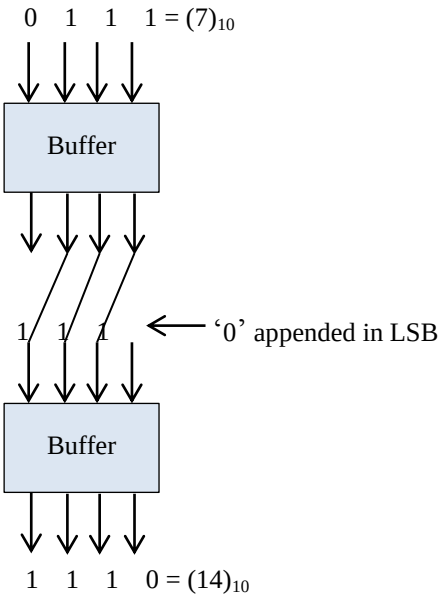

Figure 5. 1 Block diagram for obtaining twice of a binary number

In this section non-restoring division architecture is used as building block for digital serial division architecture. Non-restoring division uses add/subtract-shift approach for performing the division operation. In digital design when a set of binary number is left shifted "n" times, the result produced is "2n" times of the given binary number as shown in Figure 5.1.

Hence if a binary number "0111" is shifted one bit left, then the result is "1110" which is two times of the binary number "0111". This principle is used in non-restoring division algorithm to find twice of the partial remainder R_{i-1}. Depending upon the sign value of the partial remainder $R_{i-1,}$ the divisor is added or subtracted with twice of the partial remainder. If R_{i-1} is positive, then the divisor is subtracted from $2R_{i-1}$ else it is added with $2R_{i-1}$.This iterative process of add/subtract and shift is performed to find out the final remainder and subsequent quotients in non-restoring division algorithm.

A detailed algorithm for non-restoring division is presented below [25]:

R (0):= X;

for i in 0...p-1 loop

if R (i) <0 then Q(i)= 0; R(i+1)= 2* R (i) +Y

else Q (i) = 1; R(i+1) = 2* R(i) –Y;

end if;

end loop;

Q (0) = 1-Q (0); Q (p) = 1; R= R(p);

If X>= 0 and R<0 then R= R+Y; Q=Q-1;

elsif X<0 and R>=0 then R= R-Y; Q = Q+1

end if;

where p is the number representing accuracy of fractional bits, the dividend is $X = X_n X_{n-1}\ldots X_0$, the divisor $Y= Y_nY_{n-1} \ldots Y_0$, the remainder $R= R_n R_{n-1}\ldots R_0$, and the quotient $Q= Q_0 Q_1 Q_2....Q_p$. The condition for non-restoring algorithm is $-Y<X<Y$ must be true. Also $2^p .X = Q.Y+R$ should be satisfied, where $-Y\leq R<Y$.

The following example for calculation of -12/15 for an accuracy of 8-bits (so p=8) illustrates the non-restoring division algorithm [25].

Here R(0) = -12

Q(0)=0, R(1)= -24+15=-9

Q(1)=0, R(2)= -18+15=-3

Q(2)=0, R(3)= -6+15=9

Q(3)=1, R(4)= 18-15=3

Q(4)=1, R(5)= 6-15=-9

Q(5)=0, R(6)= -18+15=-3

Q(6)=0, R(7)= -6+15=9

Q(7)=1, R(8)= 18-15=3

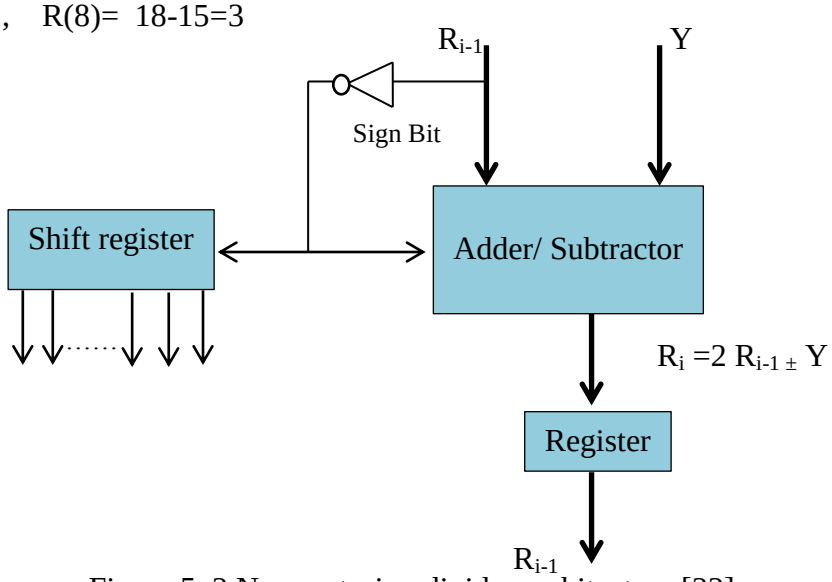

Figure 5. 2 Non-restoring divider architecture [22]

The non-restoring divider architecture is given in Figure 5.2. The architecture uses adder/subtractor, shift register and registers. Depending on the radix of the number system, the adder /subtractor cell may include ripple carry adder (RCA) or carry look ahead adder (CLA). For lower radices, carry ripple adder is suitable .For higher radices carry look-ahead adder or carry select adder is preferred. The adder subtractor cell is used to perform the addition or subtraction depending upon the quotient of the previous operation. The shift register is used to shift the partial remainder R_{i-1} to get 2 R_{i-1}.

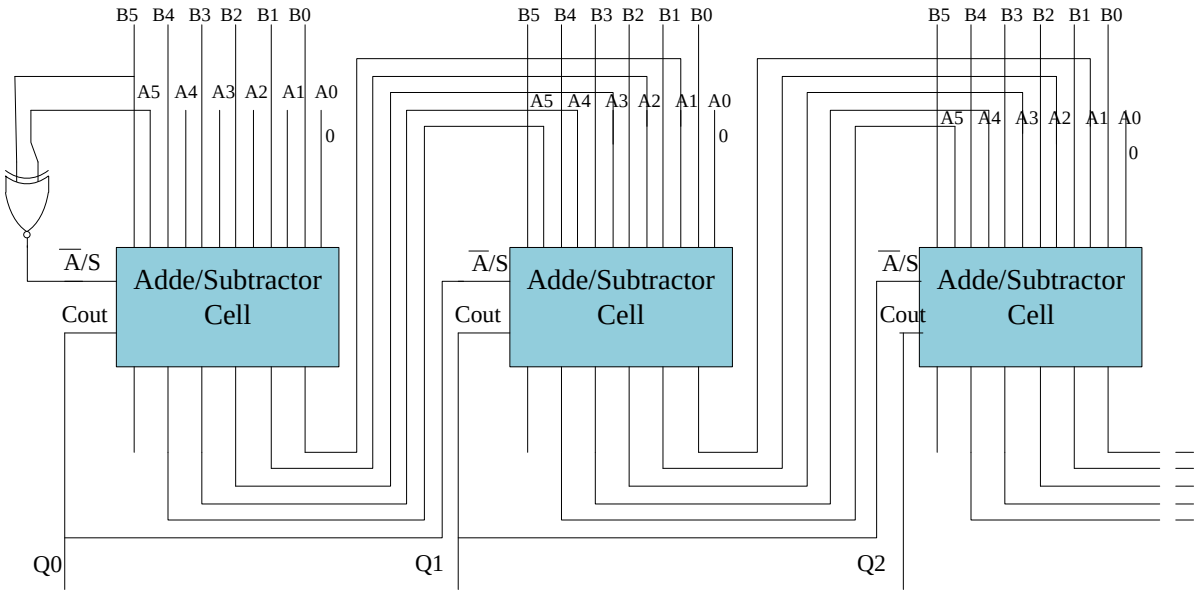

Figure 5. 3 Serial divider architecture

5.4 Architecture of Serial Divider Block

A serial divider architecture is shown in Figure 5.3. The concept is being taken from digital serial paper [24] and modified by adding one extra adder/subtractor block at the starting to get the initial remainder value R(0)=X. The architecture is shown for a 6-bit divider which can be extended for any number of bits. The architecture consists of controlled adder/subtractor cell for 6-bit operation. The adder/subtractor cell consists of six full adders and six XOR gates as shown in Figure 5.4. The operands of the cell are X and Y , where X is given directly and Y is given as one of the input of XOR gate whose another input is the select bit ($\overline{A}$/S).The select bit ($\overline{A}$/S) is used to perform addition or subtraction by the controlled adder subtractor cell. When ($\overline{A}$/S) is ‘0’ the cell performs addition, when it is ‘1’

the cell performs subtraction. All the negative numbers are represented in 2's complement form for ease of calculation of division operation. Here dividend is X, divisor is Y, quotient is Q and remainder is R. The first adder/subtractor cell is used to assign the value of dividend X to R (0). Depending upon the sign bit of R(0), Q(0)is calculated. The carry output of the first adder/subtractor cell represents the Q (0) value. The $\overline{A}$/S bit is connected to the output of XNOR gate whose two inputs are the most significant bits (MSB, sign bit) of dividend X and divisor Y. If both the X and Y are positive, then the $\overline{A}$/S is equal to 1 and a subtraction operation will be performed in the first adder/subtractor cell.

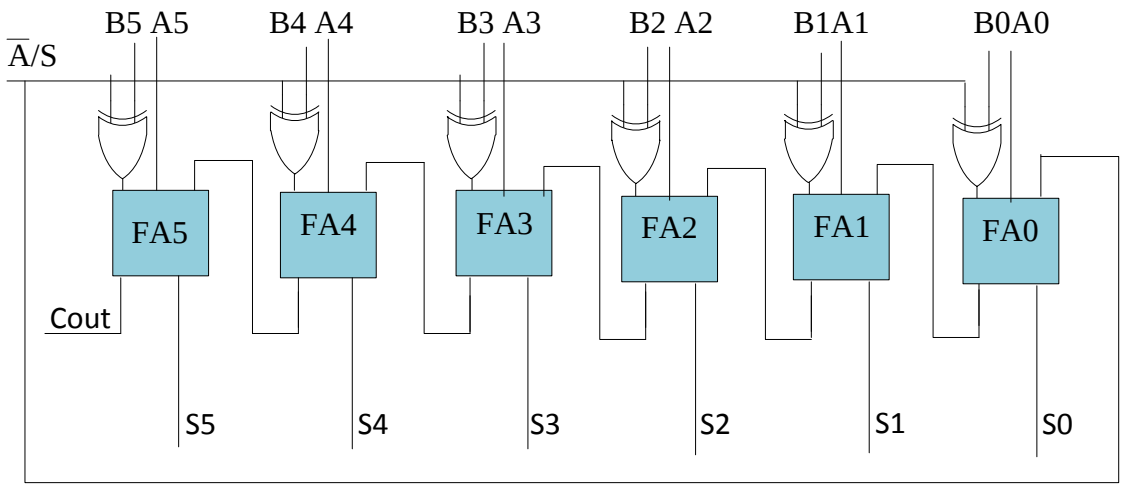

Figure 5. 4 Basic adder/subtractor cell

As described above, when quotient bit is '1', the operation is subtraction and when quotient bit is '0' the operation performed is addition. Hence the carry out of the first cell which is also the quotient bit is connected to the $\overline{A}$/S bit of adder/subtractor cell of the next cell. Hence ($\overline{A}$/S) signal of (i)th adder/subtractor cell is connected to the quotient bit of the $(i-1)^{th}$ cell and carry out bit becomes the quotient bit of the operation. The partial remainder, R(0) of the first cell is shifted one bit left in hardware connection and connected as one input to the second stage of adder/subtractor cell and a '0' is appended in the least significant bit of the next cell. This process generates 2R(0) which is further added/subtracted with divisor Y to find the next partial remainder R(1).The carry out bit of the cell becomes the quotient bit Q(1).The above connection is repeated in further stages of serial divider architecture. For p=8, nine stages are used for -12/15. Non-restoring divider may require correction circuit to

calculate the final value of quotient. However this divider architecture supports non-restoring division architecture without correction circuit taken into account. The implementation of the divider architecture is done in cadence 90 nm technology and verified for functional simulation of the architecture, which is described in the next section.

5.5 Implementation of Serial Divider Architecture

The implementation of specified architecture of serial divider using 6-bit adder/subtractor module is done using CADENCE 90nm technology library. The basic building block of the serial divider block is adder/subtractor cell. Hence adder/subtractor cell is designed using six full adders and six XOR gates. A full adder circuit is designed using transistors as shown in Figure 5.5.

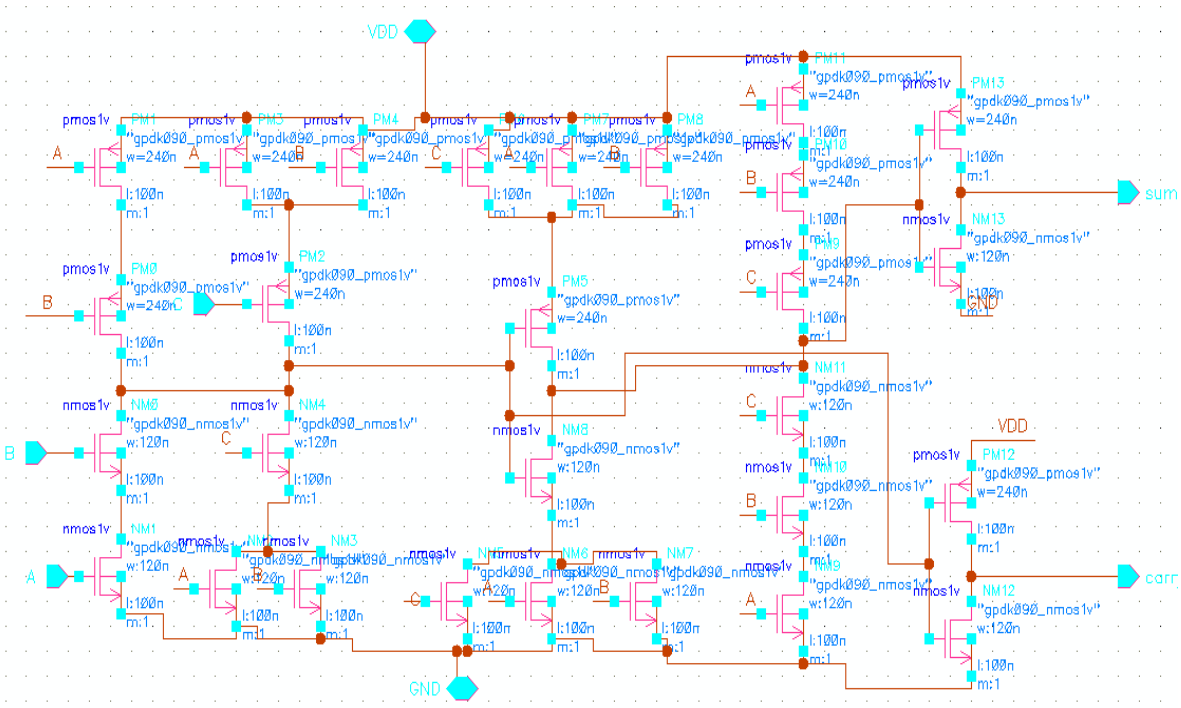

Figure 5. 5 Schematic diagram of full adder

The block diagram of basic adder/subtractor cell is designed by instantiating the full adder and XOR symbol. Figure 5.6 shows the block diagram of basic adder/subtractor cell.

A test circuit is designed for verifying the functionality of adder/subtractor cell as shown in Figure 5.7. Six sets of pulse are given to inputs A and B of the adder/subtractor cell to verify

the functionality. Here directly logic ‘1’ is given as select signal instead of XNOR for verifying addition operation. After verifying the functionality of adder/subtractor cell, serial divider architecture is designed in CADENCE and the functionality is verified through simulation, presented in the next section. The schematic for divider architecture is not shown here due to insufficient space for its clear visualization.

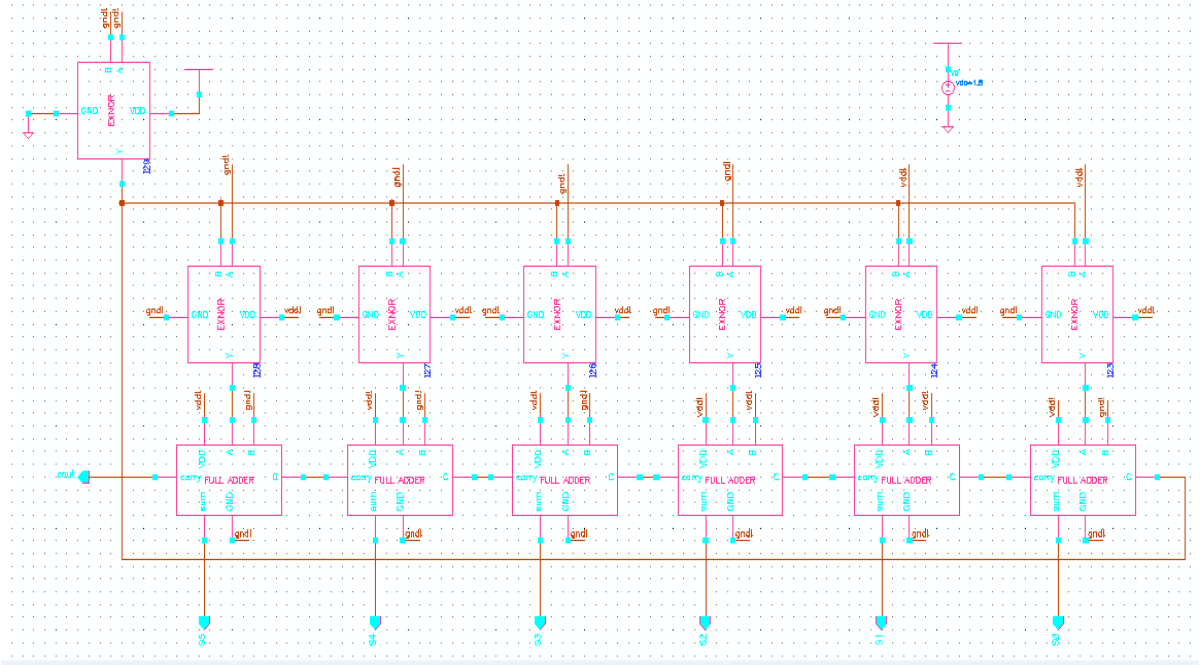

Figure 5. 6 Schematic diagram of adder /subtractor cell.

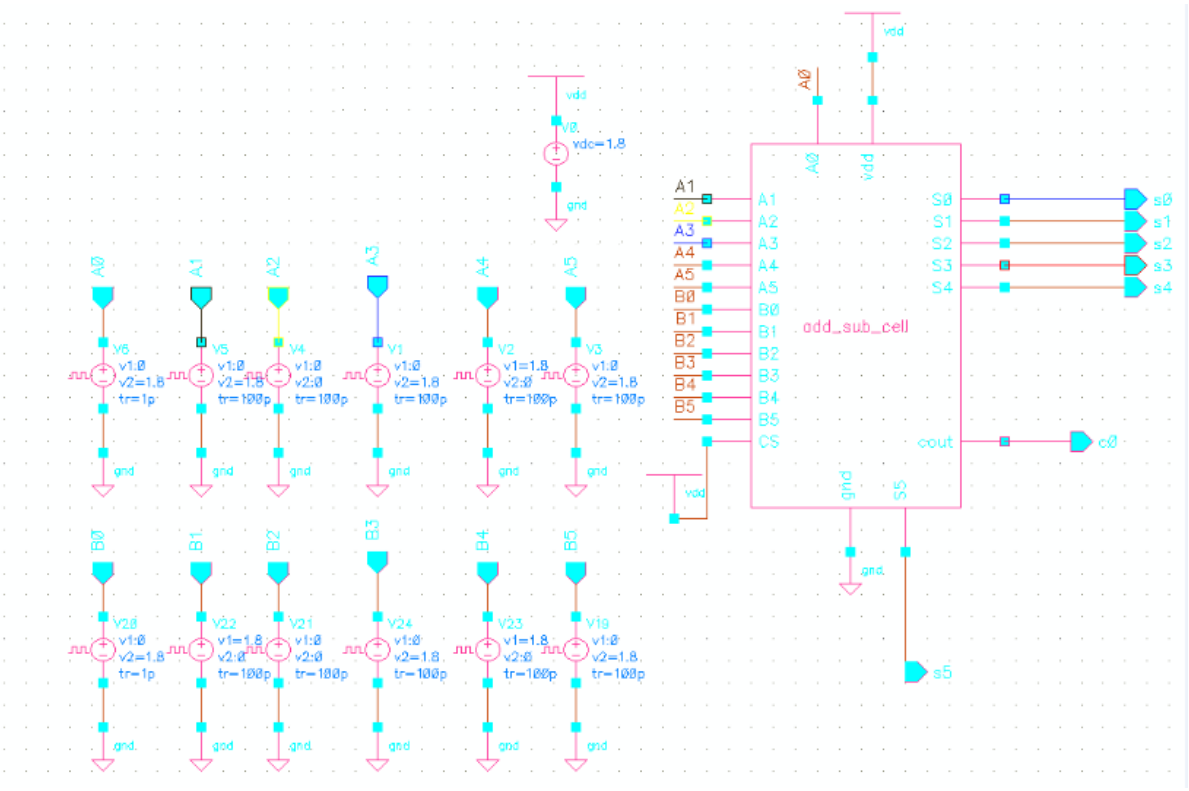

Figure 5. 7 Schematic of test circuit of adder/subtractor cell

5.6 Simulation Results of Serial Divider

The serial divider architecture is designed by instantiating the basic adder/subtractor cell. The circuit has two sets of input A and B each 6-bit wide. Hence two sets of six numbers of pulses with appropriate values are given to inputs A and B. The $\overline{A}$/S of first cell is connected to a XNOR gate whose inputs are connected to MSB of the inputs A and B. Depending upon the MSB bit the $\overline{A}$/S is evaluated and the adder/subtractor cell performs addition if $\overline{A}$/S is '0' else it performs subtraction. Each carry output of the cell gives the values of quotient and it is also connected to $\overline{A}$/S of the next adder/subtractor cell to repeat the operations. Nine adder/Subtractor cells are used to do the complete operation of -12/15 where both numerator and denominator are represented with 6-bits.

The simulation results consists of 8 sets of remainder output and 8 quotient outputs which are not possible to include simultaneously in one simulation window.Hence the simulation results are represents for inputs A and B, quotients (Q0-Q7) and final remainder R8 as shown in Figure 5.8, Figure 5.9, and Figure 5.10. The simulation is done for division values 11/23,-12/23 and 11/16 and the results are presented.

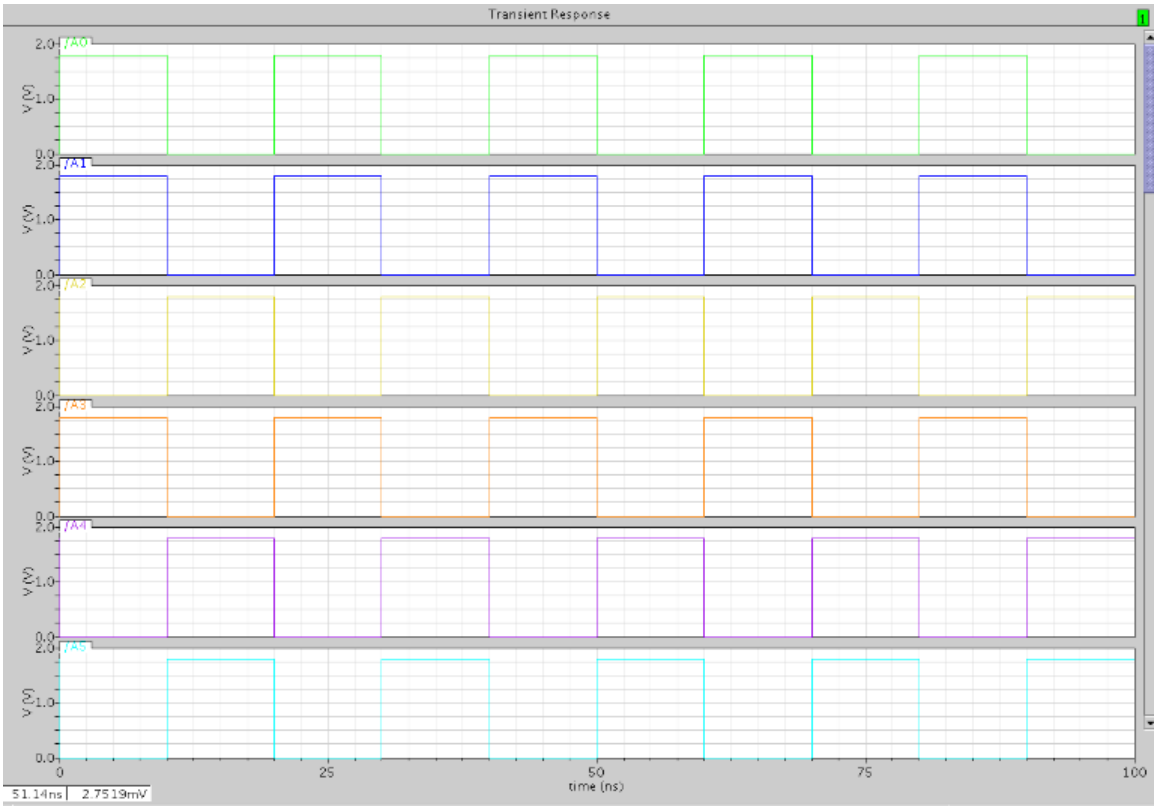

Figure 5. 8 Simulation waveform for input A

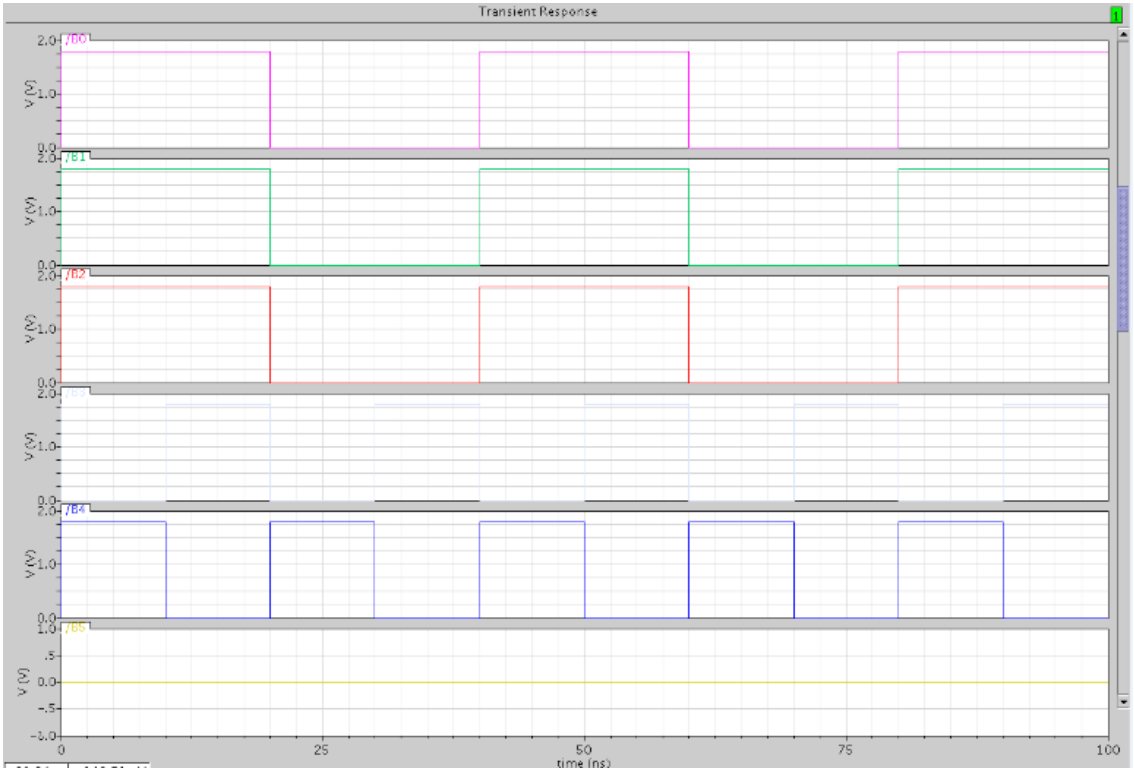

Figure 5. 9 Simulation waveform for input B

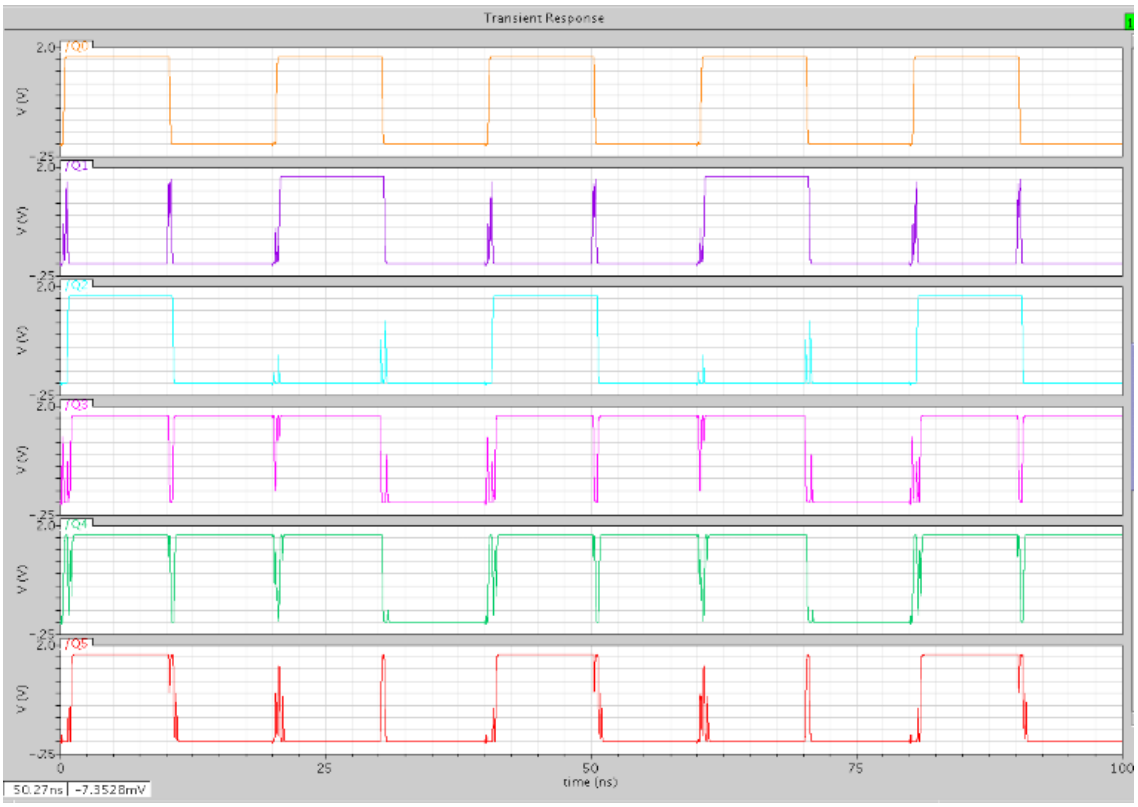

Figure 5. 10 Simulation results for quotient Q0-Q5

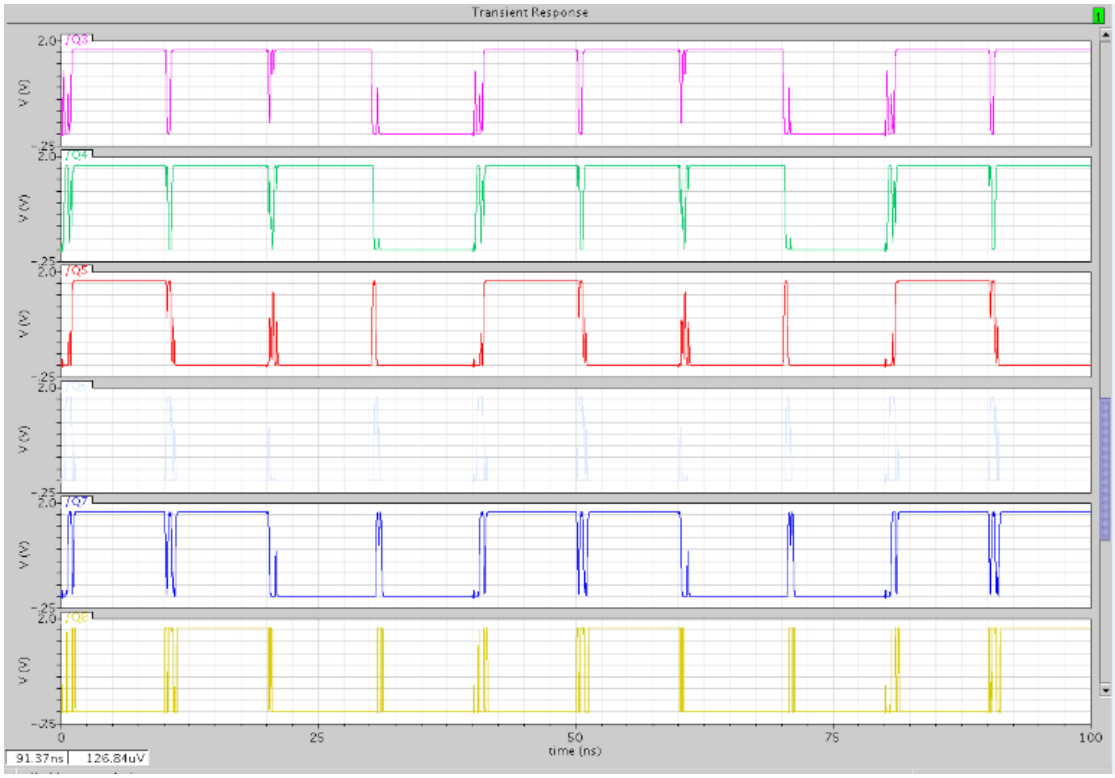

Figure 5. 11 Simulation results for quotient Q3-Q8

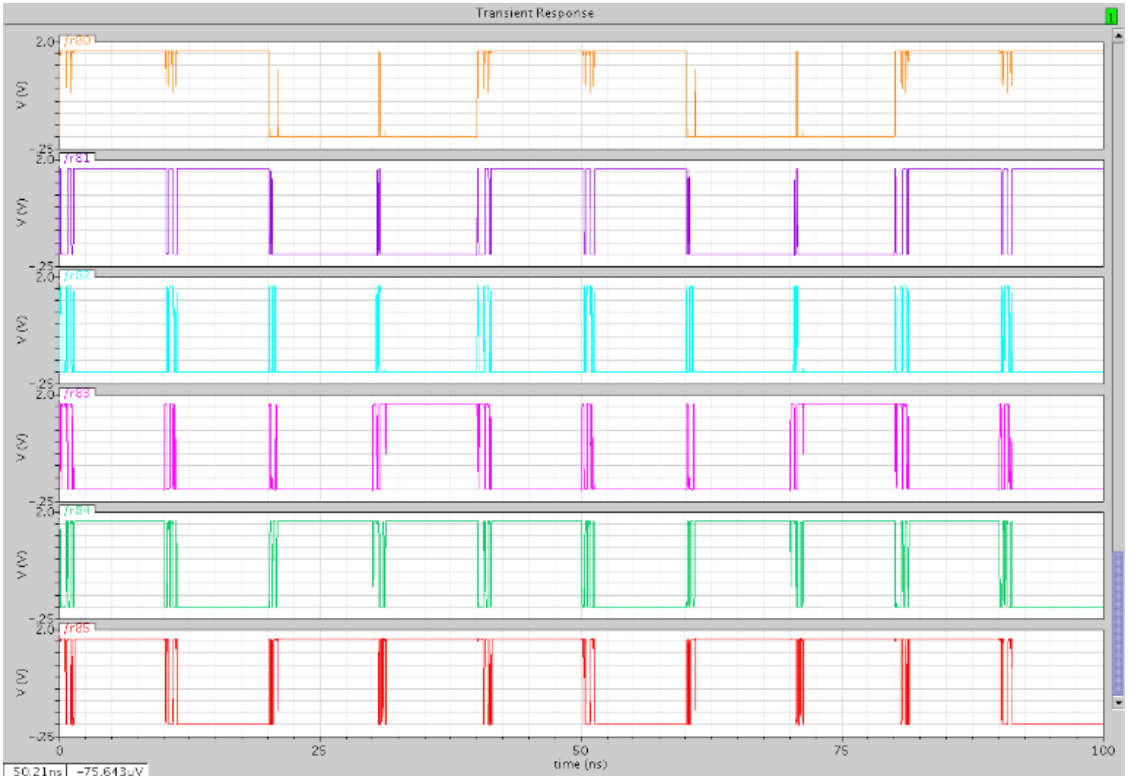

Figure 5. 12 Simulation results for final remainder R8

5.7 Conclusions

As divider is the basic building block in the VLSI architecture for algorithms used for image and computer vision applications, in this chapter, a serial divider is designed to understand the division operation and its functionality. The implementation is done in transistor level by using CADENCE 90nm technology. The algorithm used is a non-restoring type algorithm. The basic building block of the architecture consists of an adder/subtractor cell. The simulation is carried out for different division operations such as 11/23,-12/15 and 11/16 and the simulation results are presented.

The results for -12 / 15 are presented and found correct according to the example given in this chapter. The quotient values are q0=0,q1=0,q2=0, q3=1,q4=1,q5=0,q6=0,q7=1,q8=1 and the final remainder value R8=000011=3_d. Q=q0q1q2q3q4q5q6q7q8=100110011. Now 2's of Q =11001101= -205. Thus (2^8).-12 =(-205).15+3 is proved correct. After sign correction Q=-205 + 1=-204 and R= 3-15=-12. Hence (2^8).-12 = (-204).15+(-12) is proved correct. The power dissipation calculated for the divider was found to be 0.875 mW.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Signal processing based algorithms when implemented on an FPGA platform, come across several constraints like resource utilization, timing constraints, power and speed. But these constraints can be overcome by using a suitable logic or algorithm to meet the problem statement. The algorithm used in a particular system defines the VLSI architecture of the system to be designed. Similarly, in this research project, mean-shift based video processing algorithm is used for better performance of the object tracking system. Mean-shift technique is advantageous because it tracks an object for a longer time and also its simple algorithm leads to less complex logic blocks and thus reduced hardware complexity. Both VLSI architectures explained in chapter 3 and 4 are based on mean-shift algorithm and have simple logic blocks. These architectures are designed and verified successfully using VHDL. Architecture - I has less arithmetic blocks as compared to architecture –II but the later one gives more accurate tracking results as compared to the former one. Both the architectures have division block as an important part of the total blocks and thus a serial divider module is implemented successfully in 90 nm technology using cadence tool. This paves the way for further optimizing the division algorithm which can result in better architecture in terms of power, speed and area.

6.2 Future Work

The algorithm and architecture can be extended to any size of the input matrix for any video. Hardware complexity should be optimized in all aspects to meet the speed and power requirements.

A LabVIEW VI for the object tracking system can also be built for implementing this system on CRIO (Compact Reconfigurable I/O) FPGA. Again for the development of a real-time object tracking system a hardware set-up has to be done. This can be done by connecting a video capturing unit to the I/O modules of the CRIO, then by using a windows PC, the algorithm is to be executed in the LabVIEW environment and an FPGA VI can also be built to generate the bit file stream so that the algorithm is dumped on the FPGA of CRIO. Then we can display the results on a display unit connected to the I/O of the CRIO.

References

[1] Gian Luca Foresti,"Object Recognition and Tracking for Remote Video Surveillance", *IEEE Trans. on Circuits Syst. Video Tech.*, vol. 9, pp. 1045-1062, Oct. 1999.

[2] Emilio Maggio and Andrea Cavallaro, *Video Tracking*, 1st ed., John Wiley and Sons, United Kingdom: Wiley, 2011.

[3] J.Ning, L.Zhang, D.Zhang and C.Wu,"Robust mean-shift tracking with corrected background-weighted histogram",*IET. Comput.* Vision.vol.6, pp. 62-69, 2012.

[4] Alper Yilmaz , Omar Javed, "*Object Tracking: A Survey*", ACM Computing Surveys, Vol. 38, No. 4, Article 13, Publication date: December 2006. pp. 1-45

[5] D. B. Gennery, "Visual tracking of known 3-D objects," *Int. J. Comput. Vision*, vol. 7, no. 3, pp. 243–270, 1992.

[6] D. G. Lowe, "Robust model-based motion tracking through the integration of searching and estimation," *Int. J. Comput. Vision*, vol. 8, no. 2,pp. 113–122, 1992.

[7] F. Meyer and P. Bouthemy, "Region-based tracking using affine motion models in long image sequences," *Comput. Vision, Graphics, ImageProc.*: Image Understanding, vol. 60, no. 2, pp. 119–140, 1994.

[8] D. Koller, K. Daniilidis, and H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *Int. J. Comput. Vision*, vol. 10, pp. 257–281, 1993.

[9] J. Malik, D. Koller, and J. Weber, "Robust multiple car tracking with occlusion reasoning," *Eur. Conf. Comput. Vision*, Stockolm, Sweden,1994, pp. 189–196.

[10] D.Comaniciu, V.Ramesh and P. Meer, "Real-time tracking of non-rigid objects using mean-shift", *IEEE Proc.Comput. Vis. Pattern Recog.*, pp. 673-678, 2000.

[11] D.Comaniciu, V.Ramesh and P. Meer, "A robust approach towards feature space analysis", *IEEE Trans. Pattern Anal.Machine Intell.*,vol.24, pp. 603-619, 2002.

[12] Jaideep Jeyakar, R.Venkatesh Babu and K.R. Ramakrishnan, "Robust object tracking with background weighted local kernels",*Comput. Vis. Image Understanding*, vol.112,pp.296-309,2008.

[13] D.Comaniciu, V.Ramesh and P. Meer,"Kernel-based object tracking", *IEEE Trans.Pattern Anal. Mach. Intell.*,vol.25, pp. 564-577, 2003.

[14] http://www.mathworks.in/matlabcentral/fileexchange/35520-mean-shift-video-tracking.

[15] Ishtiaq Rasool Khan and Farzam Farbiz, "A backprojection scheme for accurate mean-shift based tracking", *IEEE Conf. Image Proc*., Hong Kong,2010, pp. 33-36.

[16] Jwe-Sheng Hu and Chung-Wei Juan, Jyun-Ji Wang, "A spatial-color mean-shift object tracking algorithm with scale and orientation estimation", *Pattern Recog. Lett*., vol.29,pp.2165-2173,2008.

[17] G. L. Foresti, C. S. Regazzoni, and A. N. Venetsanopoulos, "Coding of noisy binary images by using statistical morphological skeleton,"*IEEE Workshop Nonlinear Signal Processing*, Cyprus, Greece, 1995,pp. 354–359.

[18] H. Nikmehr, B. Phillips, C.C. Lim, "*A novel implementation of radix-4 floating-pointdivision /square-root using comparison multiples*", Computers and Electrical Engineering, vol. 36, 2010, pp. 850–863

[19] Cocke J, Sweeney DW, "*High speed arithmetic in a parallel device"*. Technical report, IBM Corp.; February 1957.

[20] Robertson JE., "*A new class of digital division methods"*. IRE Trans Electron Computer 1958;EC-7(3),pp.88–92.

[21] Tocher KD. Techniques of multiplication and division for automatic binary computers. Q J Mech Appl Math 1958;11,pp.364–84.

[22] Jean-Pierre Deschamps, Gustavo D. Sutter, Enrique Cantó, "*Guide to FPGA Implementation of Arithmetic Functions"*, Springer Publishers

[23] Wai-Kai Chen, "The VLSI handbook", CRC Press

[24] A E Bashagha,"*Pipelined area-efficient digit serial divider*", Signal processing, vol. 83, 2003, pp. 2011-2020

[25] Jean-Pierre Deschamps, Ge´Ry Jean Antoine Bioul, Gustavo D. *Sutter, " Synthesis of Arithmetic Circuits*", A John Wiley & Sons, Inc., Publication