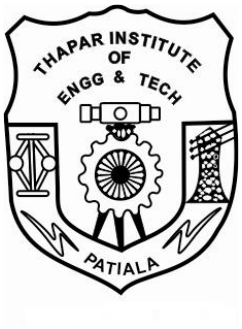
Performance Comparison of PCB Inspection using Machine Vision Algorithms in Hardware and Software

A Thesis

Submitted in partial fulfillment of the
Requirement for the award of degree of

Master of Technology
in
VLSI Design and CAD



By:
Anupam Sharma
(6040403)

Under the supervision of:
Ms Navjot Kaur
Lect. ECED Dept.

June 2006

ELECTRONICS AND COMMUNICATOIN ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA – 147004

Certificate

I hereby certify that the work, which is being presented in the thesis, entitled **"Performance comparison of PCB inspection using Machine Vision Algorithms in Hardware and Software"** in partial fulfillment of the requirements for the award of degree of Master of Technology in VLSI Design and CAD at Electronics and Communication Engineering Department of Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried out under the supervision of Ms Navjot Kaur Lecturer, ECED during January to June 2006.

(Anupam Sharma)
Signature of student

This is to certify hat the above statement made by the candidate is correct and true to best of my knowledge.

(Navjot Kaur)
Lecturer, ECED

Counter Signed

Head of Deptt.
Electronics and Communication Engineering Department
Thapar Institute of Engineering and Technology
PATIALA-147001

Dean of Academic Affairs
Thapar Institute of Engineering and Technology
PATIALA-147001

Acknowledgement

It is with the deepest sense of gratitude that I am reciprocating the magnanimity, which my guide Ms Navjot Kaur, Lecturer, Electronics and Communication Engineering Department has bestowed on me by providing individual guidance and support throughout the Thesis work.

I am also thankful to Dr. A.K.Chatterjee, P.G. Coordinator, Electronics and Communication Engineering Department for the motivation and inspiration that triggered me for my thesis work.

I would also like to thank all the staff members and my co-students who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis.

I am also thankful to the authors whose works I have consulted and quoted in this work.

Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Anupam Sharma

6040403

Index

List of Figures

Contents Page No.

List of Tables

Contents Page No.

Abstract

The human beings have very good power of vision as compared to many other species and very powerful brain to support when it comes to complete vision system of machine, there is very poor performance, in almost every aspect that man do in their daily life, the only reason is that human do not have enough computational power to process such amount of data in such a small time. Advantage that machines can provide in such situations where it takes data and quite some time to process it. One place like this is fault detection in PCBs, where a shot of PCB and check on the basis of that image whether the PCB is defective or not. This processing can be done at many levels and different image processing techniques can be used.

The software implementation of vision processing algorithm is done in the language C# .NET and uses the latest development tools like Visual Studio 2003 and the implementation of the both the algorithms for image substitution and runlength encoding in this. The results are taken by test cases made in paintbrush they are all JPEG files of different resolution.

The hardware implementation is done using VHDL and tools used are Model Sim 5.3 b and Leonardo spectrum for this purpose. The algorithm used is runlength encoding, which converts the image into gray scale then to black and white. The results of runlength algorithm are definitely better then image substitution algorithm which has been implemented in VHDL.

Chapter 1

Introduction To Machine Vision

Machine vision has progressed a lot since the early 1980s when we first starting using some of the early GE Optovision systems to sort objects by measuring area and perimeter. Today vision systems can do a lot more – just don't expect them to be as good as a human.

1.1 Image is everything

Machine vision is easy if you can get a good image into the system. Getting a good image requires a lot of patience using different optics and lighting. You have to try an endless combination of cameras, lenses, filters, lights, and some other weird and unordinary devices in order to evaluate machine vision applications. It is amazing how some applications that appear to be simple turn out to be difficult and how some difficult applications turn out to be easy. You usually never know until you sit in front of the camera with the application. What makes it so frustrating is that engineers think that if a little is a little good then a lot should be a lot of good. With optics and lighting it can be direct and sometimes it is indirect relationship. It can be very frustrating.

1.2 What makes a good vision application?

A good application is one with large differences in contrast. For example, looking at a black part on a white background, or a white part on a black background. Sometimes this can be achieved by backlighting – putting the light behind the object being viewed and viewing the silhouette of the object.

Note that there are color cameras available. So looking for distinct colors is often easy to do. Note that there are lots of ways to increase the contrast of the part of the object, which are to be viewed.

.

1.3 Components

1.3.1 Lighting / Illumination

There is a wide range of lighting techniques from placing the light in back to create a silhouette, to placing the light on the side to create shadows, to placing the light in front to illuminate the object. There are back lights, small fiber lights, light lines, darkfield illuminators, axial lighting, ring lights, collimated, etc.Fluorescent, LED, Xenon, tungsten, halogen, halide, and other types of lights.

Different applications have contrasting solutions. For example in some applications you try to eliminate all glare. In other applications to try to cause glare over a portion of the object. Some applications you try to eliminate shadows, while other applications rely on shadows. The goal for most front lighting is to produce even and consistent lighting around the object being measured. That is why ring lights are often used instead of a single source of light.

Another option is to use lasers to project one or more lines across the object. The line will allow you to measure the contours of the object. When using illumination (light) sources it is wise to pay a little more for well regulated power and light sources. Small variations in incoming voltage, frequency, or noise will cause changes in light levels making the vision project more difficult. Although most vision software has a normalized correlation (also called equalization), it is usually better to minimize variations.

1.3.2 Filters

In this case we are referring to external filters. (Software can also provide filters). Filters can include color pass, color block, polarization, UV, IR, etc. Polarizers are often used to eliminate glare and detect cracks in surfaces.

1.3.3 Lenses

There are programs available that will compute which lens is required on entering the CCD chip size, the distance from the object, and the Field Of View (FOV). There are lots of specialty lenses available such as telecentric lens, which reduce optical distortions. Minimal distortion can be achieved using higher focal length. There are accessories called "extension rings" which go between the lens and the camera that are used to help focus lenses at shorter distances.

1.3.4 Cameras

There are a wide variety of cameras on the market. The simplest is probably a “line scan” or one-dimensional camera. This camera is reading in only a single dimension and is used for reading the height or length of an object.

Standard two dimensional cameras have 512 by 512 pixels, 640 by 480 pixels, or some other similar number of pixel count. There are high-resolution digital cameras. These cameras can have resolutions of 1024 by 1024 or higher – but they are usually several times the cost of a 512 by 512 camera.

One special type of camera is the "remote head" camera. This camera is small as a tube of lipstick making them easy to mount in tight spaces. They connect via special cable to a controller that converts the image to standard video signals.

Note that some cameras have square pixels and some cameras have rectangular pixels. Preference can be achieved using square pixels while making real world measurements and often the same is used to compensate for the rectangular pixels in software. Other features of cameras include progressive scan, shutters, and built in IR filters.

1.3.5 Frame Grabber

The frame grabber is to transfer the picture from the camera into the vision processor's or computer's memory. Typically the frame grabbers that are PCI or AGP compatible are used so that they can quickly transfer images to the computer's memory for processing.

Some frame grabbers will have built-in multiplexer's. For example, the connected eight different cameras will tell the frame grabber which of the eight cameras to "grab" an image from. Some frame grabbers have a built in digital (discrete) input that act as a "trigger" to tell the frame grabber to grab an image and a digital output that fires a strobe when the frame grabber is "grabbing" the image.

Note that the latest trend is to use "Firewire" (IEEE 1394) cameras. This usually eliminates the need for a frame grabber. A color camera with 640 X 480 resolution sells for about $1,000, 1024 X 768 resolution for about $2,000 and 1280 X 960 resolution for about $3,000.

1.3.6 Vision Processor

The vision processor is a frame grabber and processor on one board. Back when computers were slow, vision processors were used to speed up vision processing tasks. Today vision processors are used less because the frame grabbers can quickly transfer to memory and the computers are much faster.

1.3.7 Computer

A lot of the image analysis is now performed on the host computer. Since the frame grabbers now use PCI and AGP computer buses and memory is very fast, images can be quickly loaded into standard computer memory where processing takes place. Most image analysis software uses the special Intel MMX and SSE instructions available in the Pentium III & IV to do many inspections quickly.

1.4 Image Analysis Software

Image analysis software can perform a wide variety of functions:

- Enhancing the image
- Blob analysis
- Pattern matching
- Optical Character

- Read bar codes and data matrix
- Perform measurements
- Overlay graphics

1.5 Accuracy

Accuracy is defined by the size of the Field Of View (FOV) divided by the number of pixels divided by the sub-pixel accuracy. For example, a 1-inch by 1 inch FOV will have a 512 by 512 pixel camera and normally the accuracy is 1 inch / 512 = 0.0019531 inches.

However, your vision system vendor really wants your business and tells you that their system uses sub-pixel measuring down to 1/32 of a pixel. Therefore accuracy is (1 / 512) / 32 = 0.000061 inches. This sub pixeling is the number one problem with suppliers and accuracy. In laboratory conditions with fake images, most good vision systems could demonstrate 1/32 sub-pixel accuracy. In the real world however, temperatures, noise, voltages and frequency are fluctuating, there are all kinds of noise, vibration, and other problems that do not allow us to achieve a 1/32 sub-pixel accuracy. Therefore, when evaluating accuracy it is necessary to find out the number used for sub-pixeling and make sure the same number is used for all systems (assuming that the every system being compared has sub pixel accuracy). Typically a factor of ½ or ¼ for sub-pixel accuracy has been assumed. In order to compare apples to apples when evaluating accuracy between systems – find out what the physical pixel count is and Field Of View (FOV) and find out if they have sub-pixel accuracy. For software that does not use sub pixel accuracy the rule of thumb is that the vision system must see two pixels. So the accuracy of your 512 X 512 camera for software that does not use sub pixel accuracy is 2 * (1 / 512) = 0.0039 inches.

The second way that some people “cheat” on accuracy is what resolution of accuracy do they use. For example if someone needs to measure down to 0.002 inches, is the 1 / 512 = 0.0019531 inches accurate enough? Some suppliers will say yes. We say no. We want to be ten times more accurate then the accuracy you need. So if you require accuracy to

0.002 inches then we are going to try to measure to 0.0002 inches so that we can tell you if a measurement is 0.0020, 0.0018, or 0.0022.

When relating pixels to real-world units such as inches, the calibration may not be the same in the horizontal direction as it is in the vertical direction. So there is a typical need to calibrate, or at least test the calibration in both X and Y directions.

1.6 Applications

There are many different applications for machine vision:

- Gauging: Measurements of length, width, area, perimeter, major and minor axes, distance
- Searching: finding parts, counting objects / features
- Alignment: X-Y and rotation, not only for the part being inspected but for a tool or subassembly that must engage the part
- Verification of presence, angle, wire color
- Pattern Matching: bleeds, pits, scratches, smears, misregistration, holes, etc.
- Inspection: are components present, properly aligned, proper size and shape
- Optical Character Recognition and Verification: printed data (such as lot and expiration date) is verified to be correct and legible, can be used to sort objects
- Bar code reading: this includes one and two dimensional bar codes. Multiple bar codes can be read from one image and the bar codes can be rotated.

1.7 Typical Programming Steps

The typical steps that are used in programming:

1. Acquire image

2. Enhance image
3. Locate image reference
4. Locate features to be checked
5. Perform measurements
6. Transfer and check results

1.7.1 Acquire Image

Image analysis starts with acquiring or "grabbing" an image. Sometimes there is a trigger (discrete input) that goes into the frame grabber telling the frame grabber to "grab" an image. This trigger could be a proximity switch that detects a new part is in position for inspection. In other cases the trigger could go into the computer and a subroutine is called to acquire the image.

Sometimes the grabbing of an image must be synchronized with the firing of a strobe light to freeze the motion of an object. Frame grabbers typically have discrete outputs for this purpose. More recently cameras have used shutters, instead of strobe lights, that open for only a brief period of time.

1.7.2 Enhance Image

The image enhancement is possible using lights, filters, and lenses external to the camera. However, a lot of operations transform the image while the image is in memory. Commonly available image enhancement tools include:

- Grab several images and average them together to remove noise
- Subtract the background to improve contrast
- Apply filters that remove noise
- Apply morphing operations that open or close (erode and dilate, thickening and thinning) objects
- Thresholding the image
- Image equalization
- Accentuate edges

1.7.3 Locate Image Reference

Typically the image is never positioned in the exact same position. Therefore a reference point is searched for to find the object in the image. Once this reference point is found (X - Y coordinates and translation) the other image properties can be referenced from this reference point. Searching for a reference point is typically time consuming so a very prominent feature on object being inspected is required. Easily distinguishable features reduce the search time and improve the accuracy of finding the orientation. Some designers add special fudical marks to the object to help locate the reference.

If there is a need to convert pixels to real-world units (inches, mm, etc.) then the fudical marks can be combined with gauges. However, more accurate gauges should be checked periodically. If the distance from the camera to lens changes, then the vision system should be able to detect scaling of the object being searched for.

1.7.4 Locate Object Features

Once it is known that where the object is located in the image, the edges can be located as also the other features of the object under inspection. Note that since the object has been searched and found, it can be known where features will be found within a much smaller area. From these features points, lines, circles, and other references to objects can be created.

These features are located in one-dimensional space whereas the previous search was done in two-dimensional space with rotation.

1.7.5 Perform Measurements

Once all of the features have been identified the measurements on all of the features of the object can be performed. Image analysis software gives you an incredible number of measurements:

- Distance

- Area
- Different measures of perimeter
- Different measures of length, width, diameters
- Measure of shape: round, roughness, compactness
- Center of gravity, mass, moment
- Angle between two lines

1.7.6 Transfer and check results

Once all of the measurements are computed, the results have to be checked against control and specification limits that were previously loaded into memory. Typically these limits were stored in a database and loaded when the program starts. The results are typically written to the hard disk, displayed on the screen for an operator to view, statistics updated, which also updates the graphs.

Chapter 2

Defects in PCBs and Types of Visual Inspection

Machine vision plays a major role in automatic manufacturing industry. Using machine vision to detect PCB defect is one of most important applications. Unfortunately, most pieces of vision equipment are abroad and expensive, which is the major motivation of this research.

This study develops a PCB inspection system to recognize PCB inner layer defects. There are 14 defect types in PCB inner manufacturing; these defects can be divided into 2 groups according to different processes. research proposes a defect outer tracing method and a pattern skeleton scanning method that can successfully detect defects after the process. In addition, using the pattern feature matching method, defects in the drilling process can be detected. The real PCB samples have been tested in the inspection system, and all defects on the sample image be detected and recognized. A fine binary image is a really important ingredient for the image subtraction operation, so this research proposes a fuzzy linguistic synthesis method for assuring the image binary operation. The method has an excellent performance to remove noise but preserve defect information. Therefore, false defects occurring at the PCB inspection system can be reduced effectively. Keywords: PCB inner layer, Defect inspection, Fuzzy linguistic method

Many important applications of vision are found in manufacturing processes, such as inspection, measurement, and some assembly operations. One of these applications is the automatic visual inspection of printed circuit boards (PCB). The circuits on PCBs are becoming much finer and more complex, so it would be a challenge for manual inspection. Comparing manual and automatic optical inspection (AOI), Moganti and AOI relieves human inspections of the tedious jobs involved[1].

(1) Manual inspection is slow and costly, leads to excessive scrap rates, and does not assure high quality.

(2) Multi-layer boards are not suitable for human eyes to inspect.

(3) With the aid of a magnifying lens, the average faultfinding rate of a human being is about 90%. However, on multi-layered boards (say, six-layered), the rate drops to about 50%. Even with fault-free power and ground layers, the rate does not exceed 70%.

(4) The industry has set quality levels so high that sampling inspection is not applicable. Production rates are so high that manual inspection is not feasible. Tolerances are so tight that manual visual inspection is inadequate.

(5) Packaging technologies become increasingly complex, substrates become more costly; hence, scrap is minimized.

2.1 Types of Defects

Figure 2-1 illustrates the more common defects found in PCBs:

FATAL DEFECTS

- Conductor breaking
- Short-circuit

POTENTIAL DEFECTS

- Pinhole
- Breakout
- Overetch
- Underetch

Fatal defects are those in which the PCB does not attend the objective they are designed for, and potential defects are those compromising the PCB during their utilization.

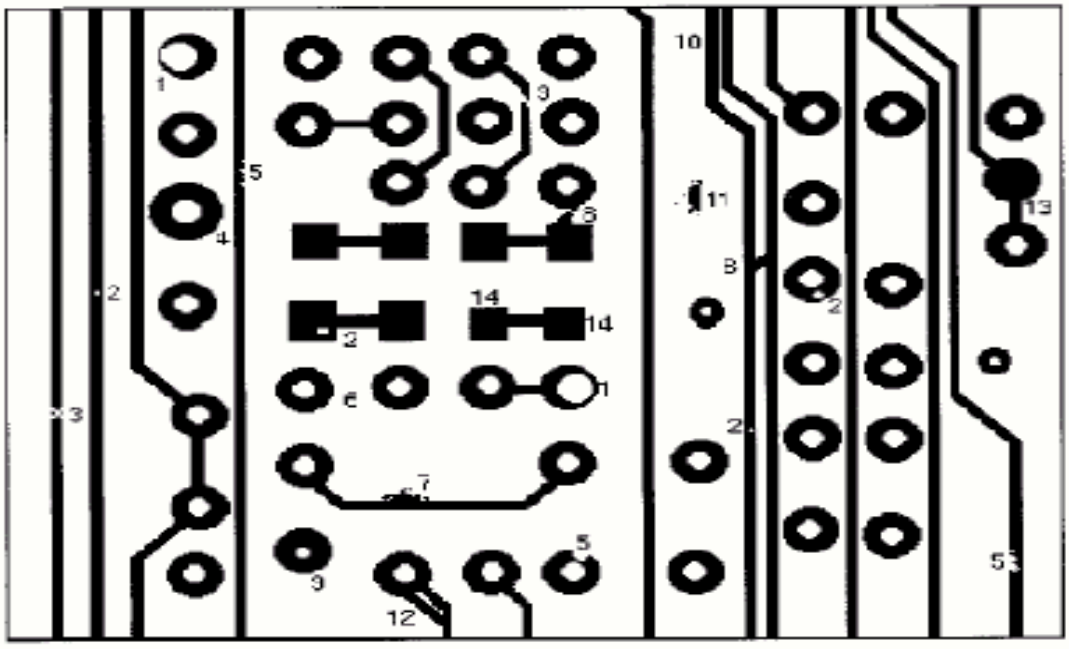


Figure 2-1 ***Description of different defects***

1. Breakout
2. Pin hole
3. Open
4. Under etch
5. Mouse bite
6. Missing conductor
7. Spur
8. Short
9. Wrong size hole
10. Conductor too close
11. Spurious copper
12. Excessive Short
13. Missing hole
14. Over etch

2.2 Vision System Algorithm Classification

Different techniques that are used for defect detection in PCBs are discussed in this section. How they are divided, what are there advantages over others etc are discussed. They are mainly divide into two parts referential and non referential, the referential are further divided into Image comparison and model based. The non referential one are divided into Morphological and Encoding based. The reference comparison methods need a standard image, which is scanned from a "golden board" or a CAD file of PCB design. The inspection board is scanned then compared to its standard image. These methods can detect defects like missing pattern, opens, and shorts easily.
The limitations of these methods are large reference image storage needed, precise alignment required, and sensitive to illumination. The non-reference inspection methods do not need any reference image or pattern to work with. These methods are also called the design-rule inspection technique. The patterns and tracks of inspected image are checked with the design specification standards. Non-reference inspection methods can avoid the disadvantages of reference methods, but these methods may miss defects that do not violate the design rules. Hybrid inspection methods involve a combination both of reference and non-reference inspection methods. These methods have the advantages of these two techniques, but hybrid inspection methods are too complex. In following section, we will review the previous proposed research of these three image inspection methods [2][3][4].

.

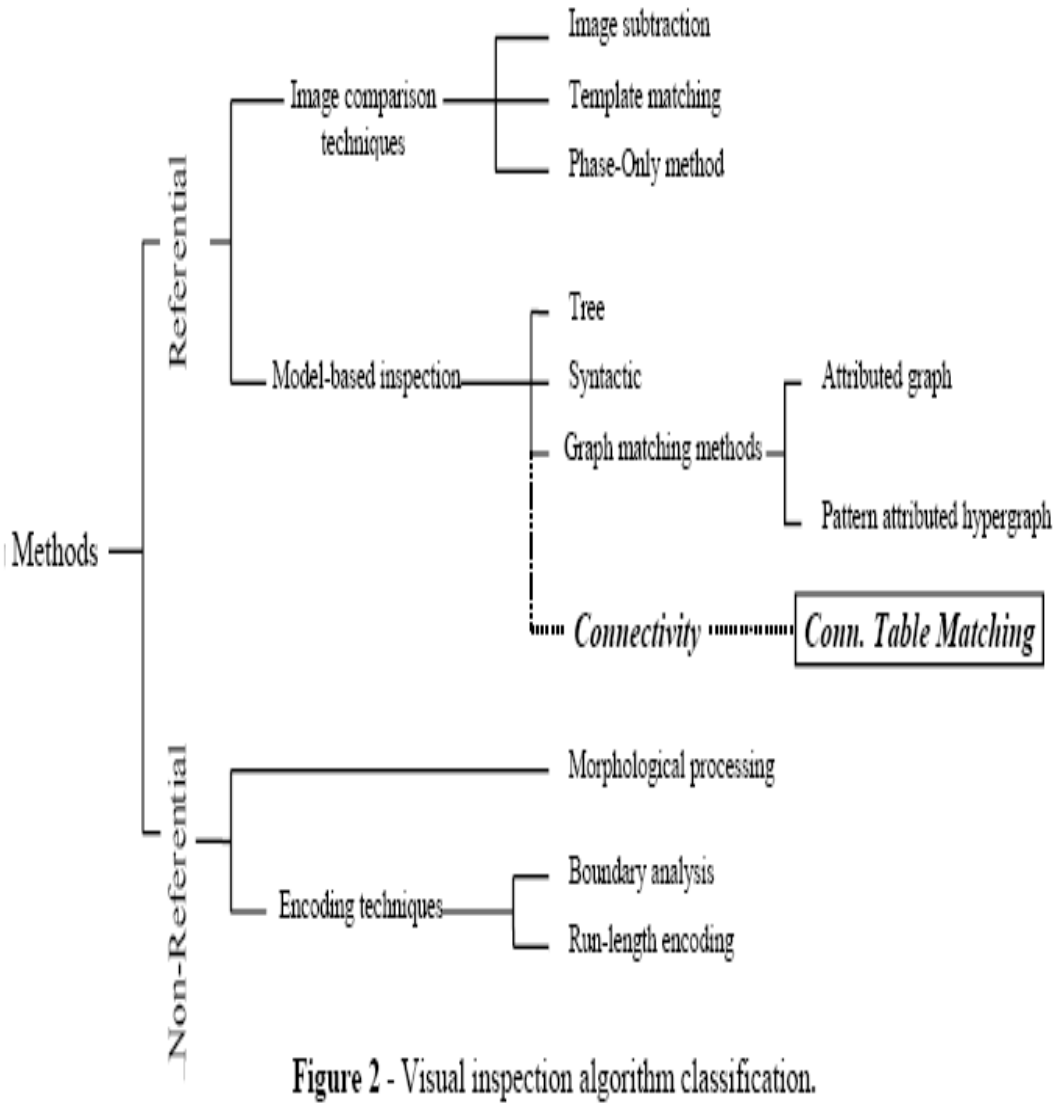


Figure 2-2 Visual Inspection Algorithm Classification

2.3 Reference Based Inspection

In image comparison techniques field, Hamada and Nakahata took inspected patterns to compare with design patterns to achieve a highly reliable inspection in 1990. They proposed two-step image processing inspection. The first step is the coarse alignment

between the detected patterns and the design patterns. The second step is the defect detection named "Local Pattern Comparison" method in which small defects up to 1.5 pixel-size can be detected without being influenced by pattern registration errors and sampling errors. Ito and Nikaido proposed a topological comparison method, which compares the inspection graph obtained from the skeletons of the conductor and insulator images of an inspection PCB with those of the standard or reference boards. Topological information incorporates weighted graphs composed of several types of nodes and edges, connections, and their locations. The defects are detected by structure character, in which is not directly doing image subtraction operation. In 1996, Wu et. al. proposed image comparison method which directly subtracts the template image form the inspection image, and then conducts an elimination procedure to locate defects in the PCB. After finding defects, all defects could be classified into one of seven defect types by three indices: the type of object detected, the difference in object numbers, and the difference in background numbers between the inspected image and template.

A structural matching method to inspect the two-dimensional visual pattern is proposed by Ja H. Koo and Suk I. Yoo in 1998. The pattern is represented by a tree where its nodes contain the polygonal boundary information derived from the pattern image. To inspect whether or not the inspected pattern is defective, the matching process first checks if the tree constructed from the pattern is isomorphic to the one constructed from the standard defect-free. If they are not isomorphic, the process stops and concludes that the pattern is defective. Otherwise, it performs the second matching step using the polygonal boundary matching function. The suggested method is illustrated for the PCB inspection and shown to have the much lower time complexity than the conventional methods using graph matching.

2.4 Non-Referential Inspection

In 1996, Borba and Facon proposed a non-reference inspection method. It can detect defects without considering a reference board. The research succeeded in verifying vertically, horizontally and 45 degrees oriented traces. In 1997, Mauro et al. proposed a novel technique for PCB inspection based on the comparison of the connected table of a

reference and a test image. The method is based on connected component analysis, which is a natural way to extract the connectivity information of the conductors of a PCB. The registration of the PCB holes, which is a common problem related to referential model techniques, is solved by the concept of zone of influence of each hole. This research describes the method and its implementation using standard morphology image processing techniques[5].

2.5 Hybrid Inspection

In 1995, Michael et al. propose a radial matching method to apply on a high volume-manufacturing environment for automatic optical inspection of multichip modules with thin films. Inspection of complex thin film metal patterns for critical defects despite high topological and cosmetic variation is discussed in their research. Madhav and Frikret proposed a hybrid inspection method in 1998. A system developed in this research can handle all of the defects simultaneously with the same approach and is significantly faster compared to the existing approaches. The system consists of three major phases: the first step is the segmentation of the golden PCB image into basic sub-patterns, the second step is the learning phase, and the third and final step is the verification/inspection phase. The system introduced the application of neural networks and fuzzy logic to PCB inspection. The method is highly parallel and works at the sub-pattern level. This study chooses image comparison techniques to be the principal inspection method for searching PCB defects. Then we use edge detect and edge-tracing process to define types of defects.

Image subtraction is the simplest and most direct approach to the PCB inspection problem. The board to be inspected is scanned and its image is compared against the image of an ideal part. Because the PCB is fabricated by CAD data nowadays, the perfect reference image is rebuilt or directly transferred from CAD data to the inspection system. Ito et. al. also performed out the defects analysis and evaluation by using PCB CAD data in 1993.In this research, two basic component data that are pad and track are obtained from a CAD file to make the ideal reference image. However, it is not always necessary for an inspection board to coincide completely with the standard board as long as some

specified design rules are satisfied. For this purpose, this study modifies ideal images that have reasonable tolerance for image subtraction process by PCB design rules

Chapter 3

Machine Vision System Architecture

Modern machine-vision systems typically contain faster and more powerful PC platforms, robust 32-bit operating systems, and easy-to-use integrated software applications, making machine-vision systems more powerful, easier to program, and less expensive to use than ever before. To take full advantage of these powerful systems and painlessly integrate them into your manufacturing line, it is best to take some time to learn the basics about what makes up a vision system, how it is implemented, and the importance of proper planning.

Machine vision can be used in a wide variety of manufacturing operations for repetitive inspection tasks in which accuracy and reliability are important (e.g., verifying date codes on food packaging, inspecting automotive parts for proper assembly, and performing robotic guidance for pick and place operations).

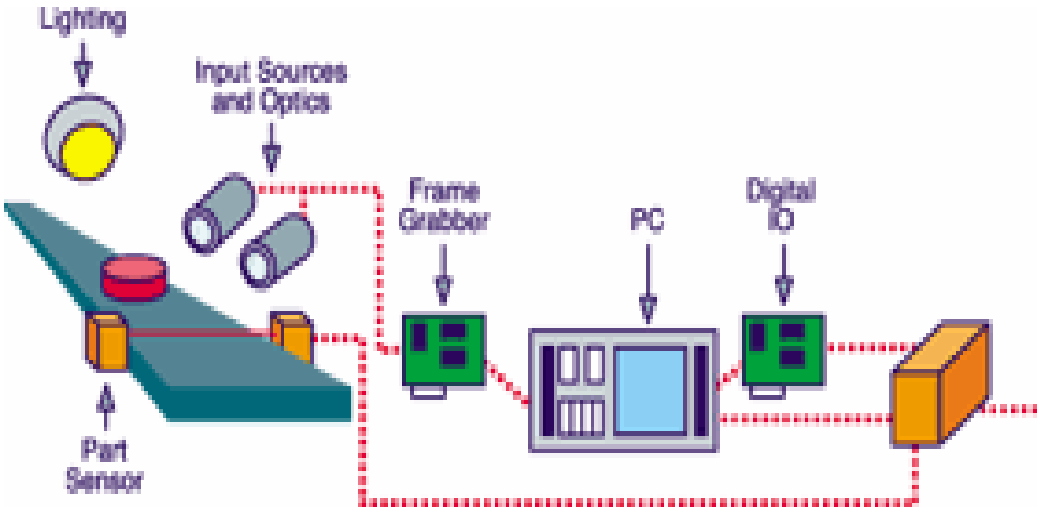


Figure 3-1 ***A Typical machine-vision.***

3.1 Building the system

Because the uses of machine vision are so diverse, specific components can vary from system to system. However, most systems generally include an input source, optics, lighting, a part sensor, a frame grabber, a PC platform, inspection software, digital I/O and a network connection, and an X-Y positioning table (see figure).

The input sources and optics usually consist of one or more cameras and optical systems that take one or more images of the part being inspected. Depending on the application, the cameras can be standard monochrome RS-170/CCIR, composite color (Y/C), RGB color, non-standard monochrome (variable-scan), progressive-scan, line-scan, or custom CCD arrays (used for x-ray).

Illuminating the part for optimal data acquisition requires outside lighting. These assemblies come in various shapes and sizes and are available in a variety of intensities. The most common lighting types are high-frequency fluorescent, LED, incandescent, and quartz-halogen fiber-optic.

For maximum efficiency, the system must be triggered by a part sensor to acquire an image when the part under test is in the correct position. Often in the form of a light barrier or sensor, the part sensor sends a trigger signal when it senses that a part is in close proximity.

The frame grabber, or video capture card, interfaces the imaging units to the host computer. The frame grabber takes the image data provided by the camera(s) in either analog or digital form and converts it to information for use by the host PC. This component is usually in the form of a plug-in board installed in the PC. A frame grabber can also provide signals to control camera parameters such as triggering, exposure/integration time, and shutter speed. Frame grabbers come in various configurations to support different camera types as well as different computer bus platforms (PCI, compact PCI, PC104, ISA, etc.)

A computer is a necessary part of a machine-vision system. Inspection applications generally require a Pentium III or equivalent. In general, the faster the PC, the less time the vision system will need to process each image. The vibration, dust, and heat often found in manufacturing environments frequently require the use of an industrial-grade or ruggedized PC. Software processes incoming image data and makes pass/fail decisions. Machine-vision software can come in many different forms and can be single function (designed only for one purpose like LCD inspection, ball grid array inspection, alignment tasks, etc.) or multi-function (designed with a suite of capabilities including gauging, bar code reading, robot guidance, presence verification, and so on).

Once the system has acquired an image, that image and the resulting data may need to be accessed by other users/systems to control the manufacturing process, communicate pass/fail information to a database, or both. Usually, a digital I/O interface board and/or network card makes up the interfacing through which the machine-vision system communicates with other machines, systems, and databases.

The X-Y positioning table automates the process of acquiring images of multiple samples. The table moves a predetermined distance after each image is acquired to properly position the next object or specimen in relation to the camera. The majority of X-Y tables have rapid smooth movement that minimizes image distortion and virtually guarantees a clean image without the need for using a progressive scan frame grabber.

3.2 Choosing Your Hardware

Careful planning and attention to detail will help ensure that your inspection system meets your application needs. Knowing your goals is perhaps the most important step in the process. Inspection operations fall into several categories: performing measurements or gauging, recognizing and identifying specific features (pattern matching), reading characters or encoded (bar code) information, detecting the presence of an object or marking, comparing objects or matching an object to a template, and guiding a machine or robot.

The inspection process can contain one operation or many depending on the requirements and goals. Firstly, it should be identified what tests are needed to adequately inspect the part (measurement, presence verification, or optical character recognition), depending on type of defects that are expected to occur. To identify which are more important, a weighted list of required and optional tests is created. Build a list that satisfies the main inspection criteria. More tests can be added, but keeping in mind that more tests means more inspection time.

Knowing your speed requirements for testing is critical. Knowing the amount of time the system will have to inspect each component or part will not only determine the minimum clock speed of the PC but may also affect the speed of the line. Many machine-vision software packages incorporate a clock/timer so that each step of the inspection operation can be closely monitored. From this data, you can modify the program and/or the motion process of the part to fit within the required timing window. Often, PC-based machine-vision systems can inspect 20 to 25 components per second, depending on the number of measurements or operations required and the speed of the PC used.

A machine-vision system is only as strong as its individual components. Any shortcuts made during the selection process—especially those involved in the optics and imaging path—can greatly reduce the effectiveness of a system. The following are a few basics you should keep in mind when choosing components involved in the image path.

Camera selection is directly tied to the application requirements and usually involves three main criteria: whether the test involves a color-specific imager, whether the parts will be in motion, and what resolution will be required. Monochrome cameras are used for a majority of inspection applications because monochrome images provide 90% of the available visual data and are less expensive than their color counterparts. Color cameras are used when inspection applications require color-specific image data to be analyzed. Whether the part being inspected is stationary or in motion will dictate the exposure time and whether a standard interlaced camera can be used or if a progressive-scan camera is required to ensure a sharp image. In addition, the camera's resolution should be high enough to ensure that it can capture the proper amount of information needed for the

inspection task. Finally, cameras should be high quality and rugged enough to withstand vibration, dirt, and heat present in an industrial environment.

Optics and lighting are often overlooked. When poor optics or lighting is used, even the best machine-vision system will not perform as well as a less capable system with good optics and adequate lighting. Design properly from the beginning, and don't depend on software to overcome design flaws. The typical goal for optics is to use lenses that produce the sharpest clarity and largest usable image, thus providing the best image resolution. The goal for lighting is to illuminate the key features being measured or inspected. The type of light used often will be dictated by those features, which include the color, texture, size, shape, and reflectivity of the object.

Although the frame grabber is only one part of a complete machine-vision system, it plays a very important role. The choice of frame grabber is defined by the characteristics of the camera it must interface to—for example, is it monochrome, color, digital, analog, and so on. With digital frame grabbers, the goal is to ensure that the digital image data from the camera is formatted properly prior to passing it onto the PC for processing. With analog frame grabbers, the goal is to acquire the image data from the camera and convert it to digital data with as little alteration to the image data as possible.

Using the wrong frame grabber can introduce errors in the image data. Industrial frame grabbers are typically used for inspection tasks, for example, while multimedia frame grabbers should be avoided. Multimedia boards can alter the image data with automatic gain controls, edge sharpening, and color-enhancement circuits. Although the images look more appealing to the eye, when processed by the system's software they can result in errors that directly affect the accuracy of the inspection process.

Removing as many variables as possible is key. The human eye and brain can identify objects in a wide variety of conditions. A machine-vision system is not as versatile; it can only do what it has been programmed to do. Knowing what the system can and cannot see will help you avoid false failures (wrongly identifying good parts as bad) or other inspection errors. Common variables to consider include ambient lighting, background

color, requirement for image focus, and large changes in part color, finish, orientation, or position. Proper camera mounting, secure lighting positions, constant and repeatable part/component positioning, and blocking of external or surrounding lighting can eliminate many common set-up and false-failure problems.

New advanced algorithm technologies for dynamic machine-vision applications are emerging that can perform inspection applications without the need for strict control of operating parameters. These algorithms are designed to allow the system to function efficiently in environments that dynamically change. Machine-vision systems that use these new algorithms are no longer hindered by changes in part orientation, rotation, scaling, lighting, image quality, and so on. A machine-vision package with dynamic algorithms, such as our DT Vision Foundry package, can cope with changing parameters. For example, a search tool based on this technology is well suited to applications in which lighting variations, such as poor contrast, glare, reflection, and inversion, are problematic.

3.3 Choosing Software

The machine-vision software is the centerpiece of the inspection system. The software selected will determine the length of time required to generate and debug inspection programs, what inspection operations can be performed, how well those operations can be performed, as well as many other important factors (see sidebar below).

Machine-vision software packages that provide a graphical user interface are usually easier to program than their code (Visual BASIC or Visual C++) counterparts but can sometimes be limited when specialized features or functions are required. Although they require programming expertise, code-based packages can be more flexible if you're developing complex application-specific inspection algorithms. Some machine-vision software packages offer both graphical and code-level development environments, providing the best of both worlds and giving users the additional flexibility of selecting the environment needed to match the application requirements as well as the programming expertise available.

The overall objective for a machine-vision system is to perform a quality assurance role by separating the good parts from the bad ones. To do this, the system needs to communicate to the manufacturing line that a part is bad so that action can be taken. Usually this information is conveyed via the digital I/O board, which is connected to the manufacturing line's programmable logic controller (PLC). The bad part is then separated from the good parts. In addition, the machine-vision system may be connected to a network to allow data to be transferred to a database for data logging purposes. Quality-control personnel to determine why the fault occurred can then analyze data.

Careful planning at this stage will ensure smooth integration of the machine-vision system into the line. Many companies would like to network their vision systems so that remote access is available. Transferring images and data can be very beneficial to the customer to prove total quality control. This information/requirement is very strategic to the design of a machine-vision system. You must know what PLC will be used and how is it interfaced, what types of output signals you'll require, what kind of network is currently used or required, and what kind of file formats will be transferred on the network. Typically, communication to a database is done via an RS-232 line connected to the manufacturer's network to track failure information.

When selecting components for a machine-vision system, consider future production requirements and changes. This can directly impact which features will be needed in the machine-vision software/hardware, as well as how easily the system can be altered to meet changing requirements and different tasks. Planning ahead will not only save time but will help reduce the overall cost of the system if it can be used for other inspection tasks in the future

Chapter 4

Software Implementation

Machine vision algorithms can very easily to implement in the high level languages. The Language that have used for this purpose is C# it is language Made by Microsoft. The Developing environment used by me is visual studio 2003. The System Specification are 256 MB RAM P4 HT technology and XP operating System[13][14].

4.1 Microsoft C#.NET Language.

4.1.1 C#

C#, a Java-like programming language, was submitted by Microsoft to the ECMA standards group in mid-2000. On July 11, 2000 Microsoft had a press release announcing the .NET Framework to unite programming languages for Web-based uses. (This C# Directory began as the first C# Web site on the Internet, and has evolved as more information and references for developers has become available.)

4.1.2 C# versus Java

C# and Java are both new-generation languages descended from a line including C and C++. Each includes advanced features, like garbage collection, which remove some of the low level maintenance tasks from the programmer. In a lot of areas they are syntactically similar.

Both C# and Java compile initially to an intermediate language: C# to Microsoft Intermediate Language (MSIL), and Java to Java bytecode. In each case the intermediate language can be run - by interpretation or just-in-time compilation - on an appropriate 'virtual machine'. In C#, however, more support is given for the further compilation of the intermediate language code into native code.

C# contains more primitive data types than Java , and also allows more extension to the value types. For example, C# supports 'enumerations', type-safe value types which are limited to a defined set of constant variables , and 'structs', which are user-defined value types .

Unlike Java, C# has the useful feature that we can overload various operators. Like Java, C# gives up on multiple class inheritance in favour of a single inheritance model extended by the multiple inheritance of interfaces . However, polymorphism is handled in a more complicated fashion, with derived class methods either 'overriding' or 'hiding' super class methods. C# also uses 'delegates' - type-safe method pointers . These are used to implement event-handling. In Java, multi-dimensional arrays are implemented solely with single-dimensional arrays (where arrays can be members of other arrays. In addition to jagged arrays, however, C# also implements genuine rectangular arrays

4.1.3 About .NET

.NET (dot-net) is the name Microsoft gives to its general vision of the future of computing, the view being of a world in which many applications run in a distributed manner across the Internet.

Firstly, distributed computing is rather like object oriented programming, in that it encourages specialized code to be collected in one place, rather than copied redundantly in lots of places. There are thus potential efficiency gains to be made in moving to the distributed model.

Secondly, by collecting specialized code in one place and opening up a generally accessible interface to it, different types of machines (phones, handhelds, desktops, etc.) can all be supported with the same code. Hence Microsoft's 'run-anywhere' aspiration.

Thirdly, by controlling real-time access to some of the distributed nodes (especially those concerning authentication), companies like Microsoft can control more easily the running

of its applications. It moves applications further into the area of 'services provided' rather than 'objects owned'.

Interestingly, in taking on the .NET vision, Microsoft seems to have given up some of its proprietary tendencies (whereby all the technology it touched was warped towards its Windows operating system). Because it sees its future as providing software services in distributed applications, the .NET framework has been written so that applications on other platforms will be able to access these services. For example, .NET has been built upon open standard technologies like XML and SOAP.

At the development end of the .NET vision is the .NET Framework. This contains the Common Language Runtime, the .NET Framework Classes, and higher-level features like ASP.NET (the next generation of Active Server Pages technologies) and WinForms (for developing desktop applications).

The Common Language Runtime (CLR) manages the execution of code compiled for the .NET platform. The CLR has two interesting features. Firstly, its specification has been opened up so that it can be ported to non-Windows platforms. Secondly, any number of different languages can be used to manipulate the .NET framework classes, and the CLR will support them. This has led one commentator to claim that under .NET the language one uses is a 'lifestyle choice'.

Not all of the supported languages fit entirely neatly into the .NET framework, however (in some cases the fit has been somewhat Procrustean). But the one language that is guaranteed to fit in perfectly is C#. This new language, a successor to C++, has been released in conjunction with the .NET framework, and is likely to be the language of choice for many developers working on .NET applications.

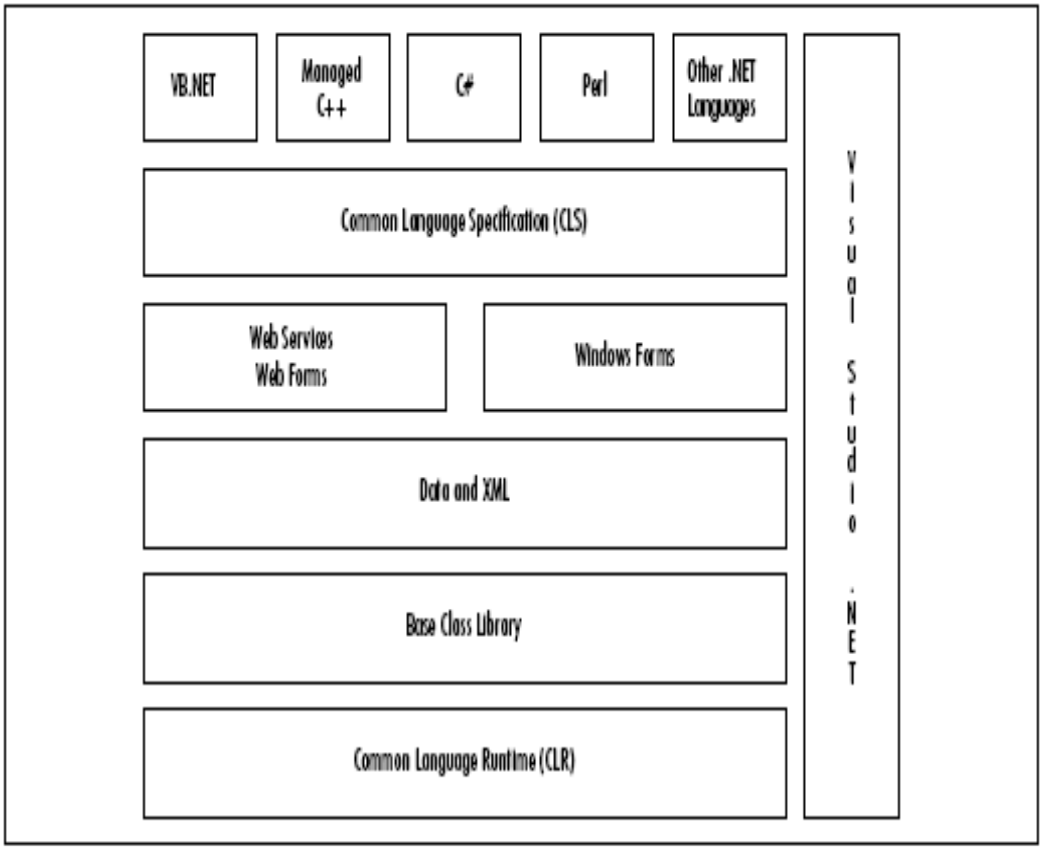


Figure 4-1 ***.NET Architecture***

4.2 Methods of implementation

4.2.1 Gray Scale Conversion

This part of software does the work of converting a image from RGB format to Gray and then Black and white. C#.net and Visual Studio provides to methods to get the pixel values of the image. So I have made the code form both the methods. They are

- GetPixel
- Unsafe code

1.GetPixel Method :

In this code I first of all created a Instance of Class Image Present in the System.Drawing namespace the and used the Abstract Class Bitmap .Then used the GetPixel Function to get the values of the pixels into the class color which has all the

three components R,G,B. Then added the value sand divided them by three to get the gray scale value of the Pixel and the placed it back into the image.
As for this purpose we used two for loops along the height and width the complexity of the algorithm becomes O(N2). That is time to convert the image into Gary time will increase in Square with the input.

Gary color = (R+G+B)/3

2.Unsafe code :

As discussed earlier All the languages under the flage of .NET use the CLR (Common Language Runtime) for managing the environment for its executable. The part of C# code that is not managed by this CLR is called the Unsafe code. In this part of code we can use pointer directly into memory and modify the memory location.

In this part of software a point is declared to each row in the image and then created. Another pointer of type byte is created in the memory and connected it to that row. Now it can access each byte in the Row and the same formula is used ,as above to convert the image into Gray scale.

4.2.2 What is the Difference Between the methods?

The performance difference between both the methods is huge the unsafe code does this very effectively as compared to other. This is also clear from my results in the end. Now the question comes why is the difference there? The reason is that, when we use the GetPixel the control is transferred to the class which intern collects data from other Class used for display.

Image Substitution method

In this method, comparing two images Back to Back to Back to Back performs the check. That is each Pixel of one image is compared with Pixel of other image. This is a good method but in ideal Conditions but when it comes to real Conditions this method performs very badly. Let us take one example in the images below.

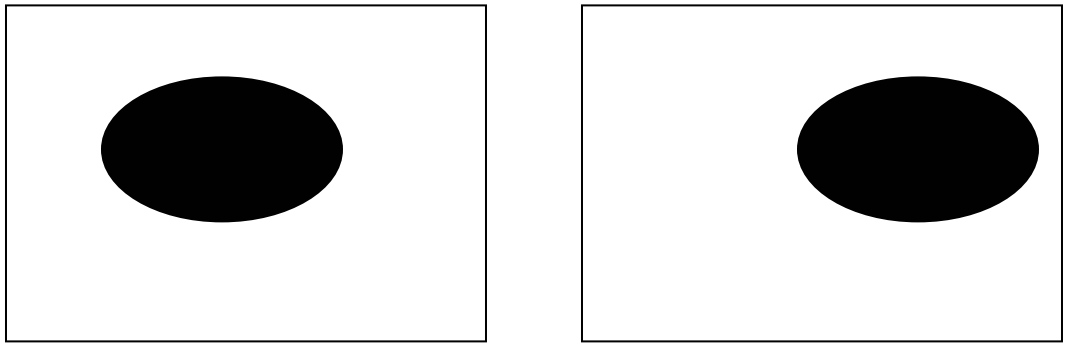


Figure 4-2 ***Disadvantages of image substitution Method***

These both are almost similar images but just have been displaced a little. When the image substitution algorithm is used, it makes pixel-by-pixel based comparison and will give the result that these Images are different. The point here is that the algorithm does not perform good if the Images have been displaced a little

So the use of this algorithm for the purpose of PCB comparison can make the results to be drastic. Because as the PCB is little dislocated while taking the image the pixel won't match and result will be faulty.

To avoid this there are many techniques like template matching etc but a different approach called Runlength encoding can be implemented which provides much better Results than Image substation method in terms of both Time and Freedom to move the image.

4.3 Runlength Encoding

In this technique a count on type of Pixels in the rows of the image in continuity and is compared with the same of the correct image. The principle of runlength encoding has been explained as follows:

.

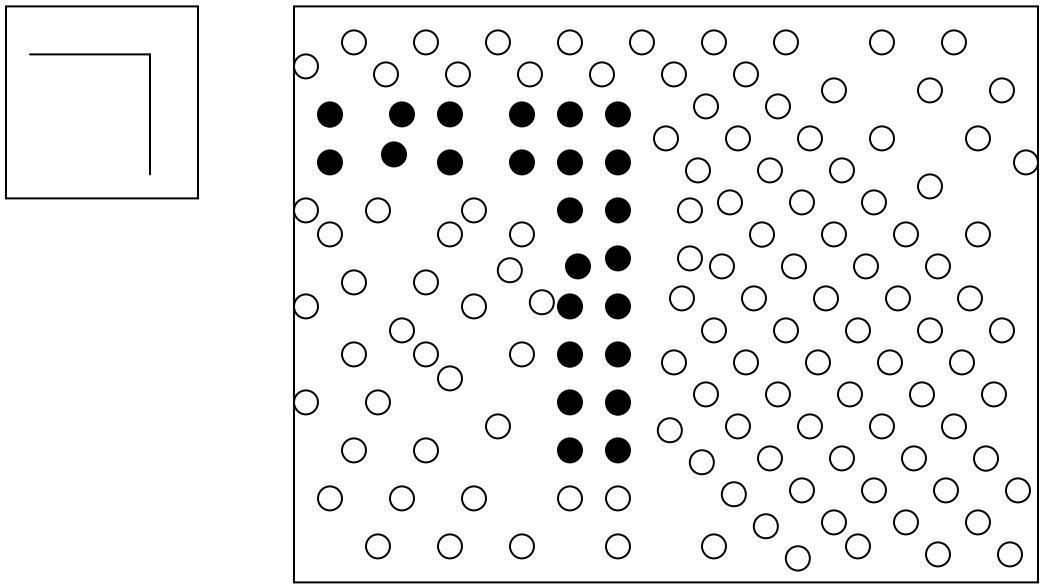


Figure 4-3 ***Pixel level view of line.***

In this image scan the rows of the pixels and find out the color of the pixel whether if it is of color BLACK. We initialise the variable to one and then count the number of BLACK pixels in continuity and then store them.

For the above example the file will look this

BLACK 6

BLACK 6

BLACK 2

BLACK 2

BLACK 2

BLACK 2

BLACK 2

This is the sequence of number of the black pixels. So we first convert both the files into the Runlength files and then compare them entry wise to check that the sequence of black color are same or not. This helps in two ways

- In one comparison we compare more the one pixel hence increase the efficiency
- We don't compare the black pixels.

Using this implementation we have a tremendous reduction in comparisons about 95% (Results) and hence the speed increases

The freedom that this algorithm provides is that even if the image moves along the X or Y-axis the comparison won't result in error. And to get even better results a little error in the comparison like a value of 1 pixel is allowed.

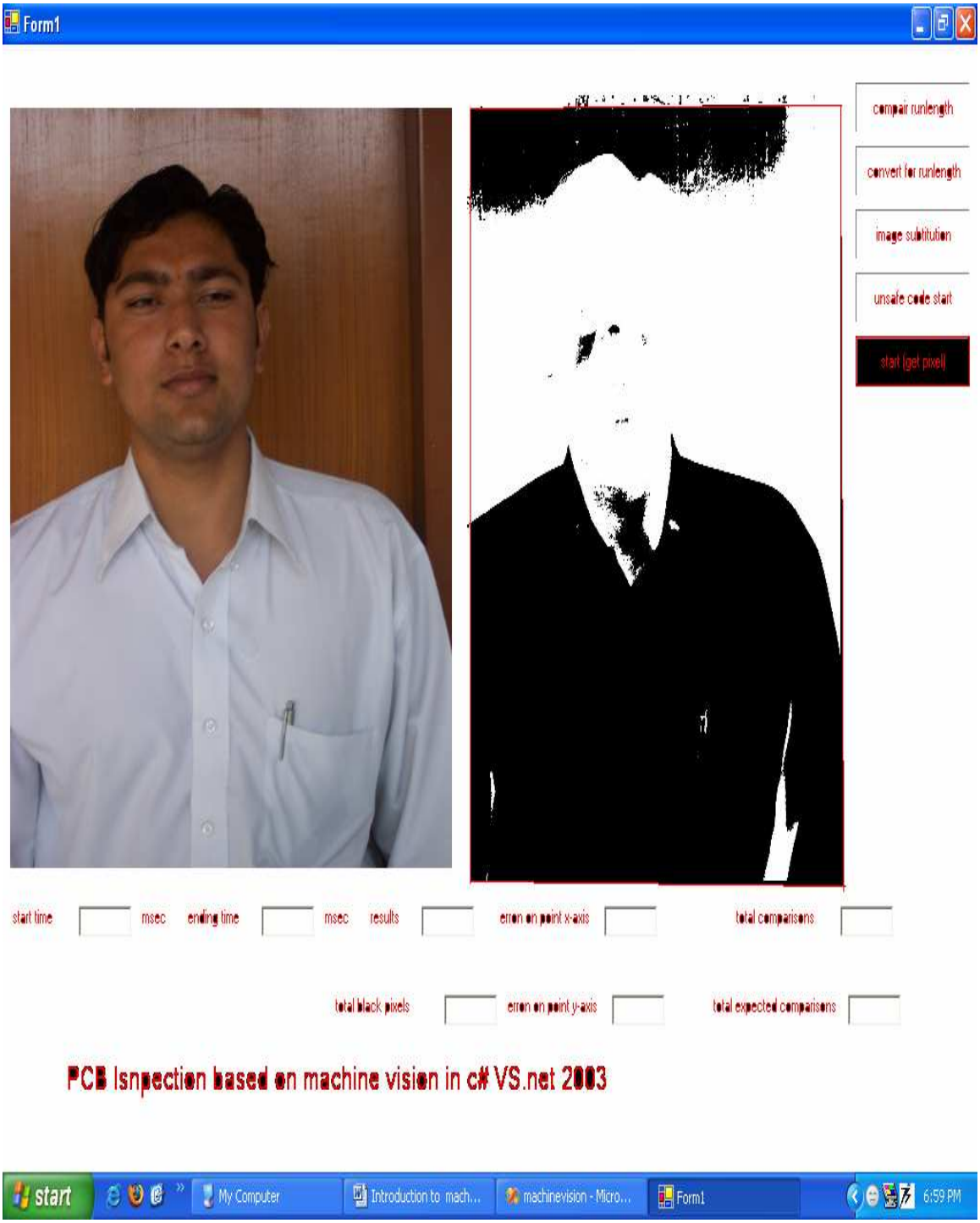


Figure 4-4 ***View of the Software***

Chapter 5

Hardware Implementation

This part of report will discuss about Run length encoding based algorithm being implemented in VHDL to get the required comparison. The entire architecture of this vision system is divided into many sub systems and signals to synchronize them. Each sub part will be discussed individually with the signaling mechanism.

There are a total of ten different entities that I have used in my set up and they are presented below

1. **Store color**
2. **Add colors**
3. **Divider**
4. **B/W converter**
5. **Count Pixel**
6. **Control unit**
7. **Address Generation Unit**
8. **Write Data**
9. **Memory interface Unit**
10. **Comparator**

These are all the entities used and some of them do have multiple instances in the architecture.

5.1 Store Color

This is the very first entity, which is used to store the 8 bit data that is coming from RAM. That is when there is a image in RGB Format and have 8 bit to each color, then there is a need to temporarily store data some where. There are three instances of this entity in which every color can be stored. The entity in VHDL has Clk, load, data, and value signals. The load signal is used as indicator when it has value as one the data from data bit vector is stored and is available from next clock cycle at output.

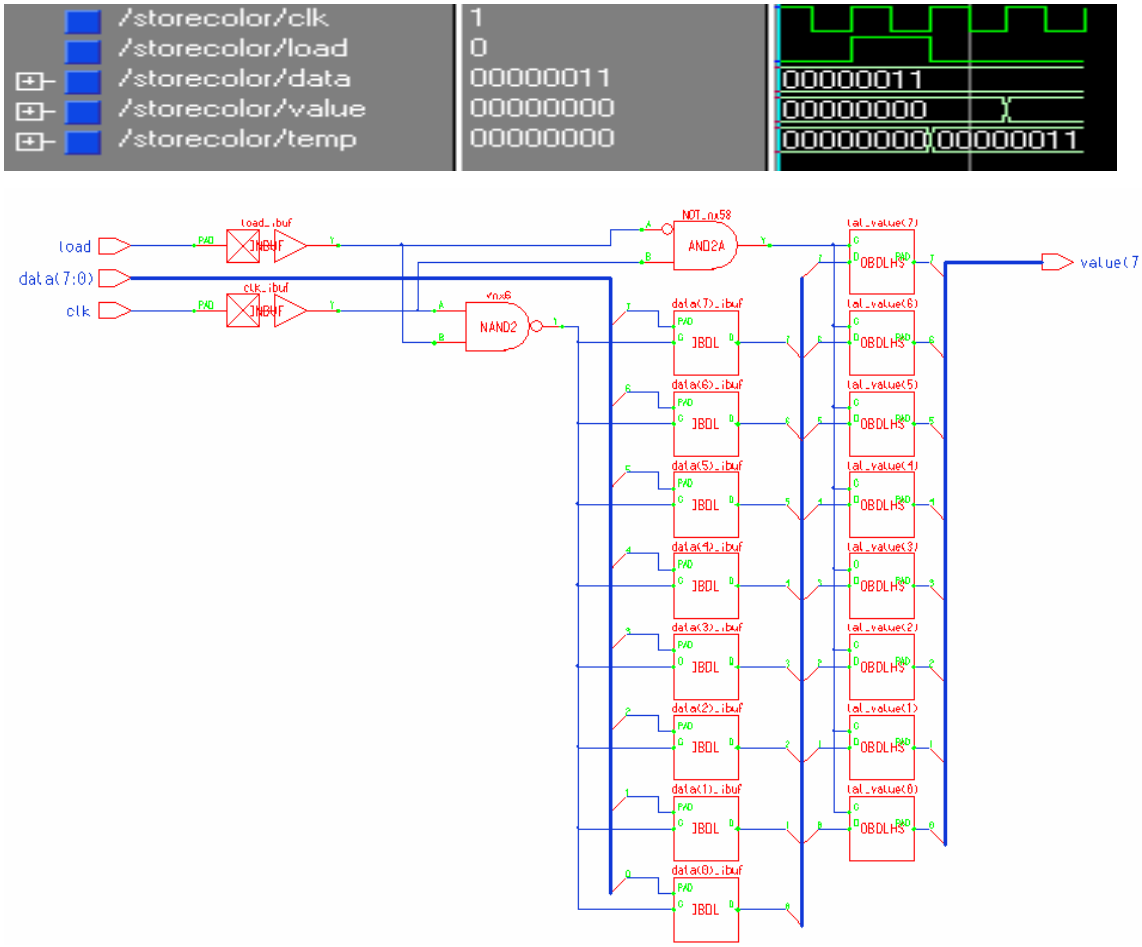


Figure 5-1 Gate level diagram of store color

5.2 Add colors

This is the entity, which adds the three colors and gives the result. This step is required because the equation for converting a pixel from RGB to gray is sum of all the colors divided by 3. The signals used for this purpose are clk, Start, red, green, blue, done and value. The array value is of bigger size then R,G,B because there sum can exceed the value of 8 bits. In order for this value to work there is a need of R,G,B and to make start high for one clock cycle. Then the value of result will come in the Value signal array.

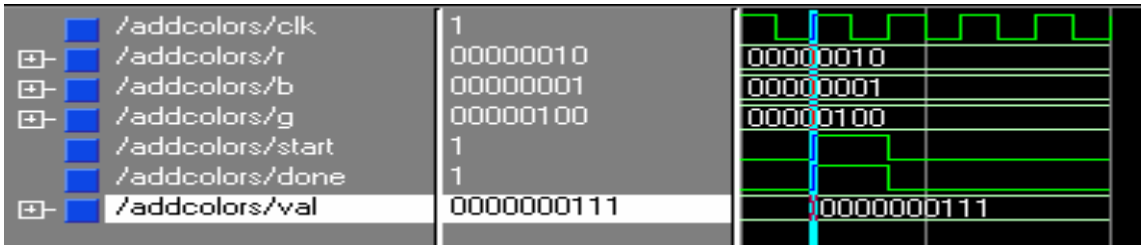


Figure 5-2 ***Timing Diagram***

The entire circuit diagram is the inter connection ckt of NAND, and , XOR and NOT gate with a layer of latches in the end.

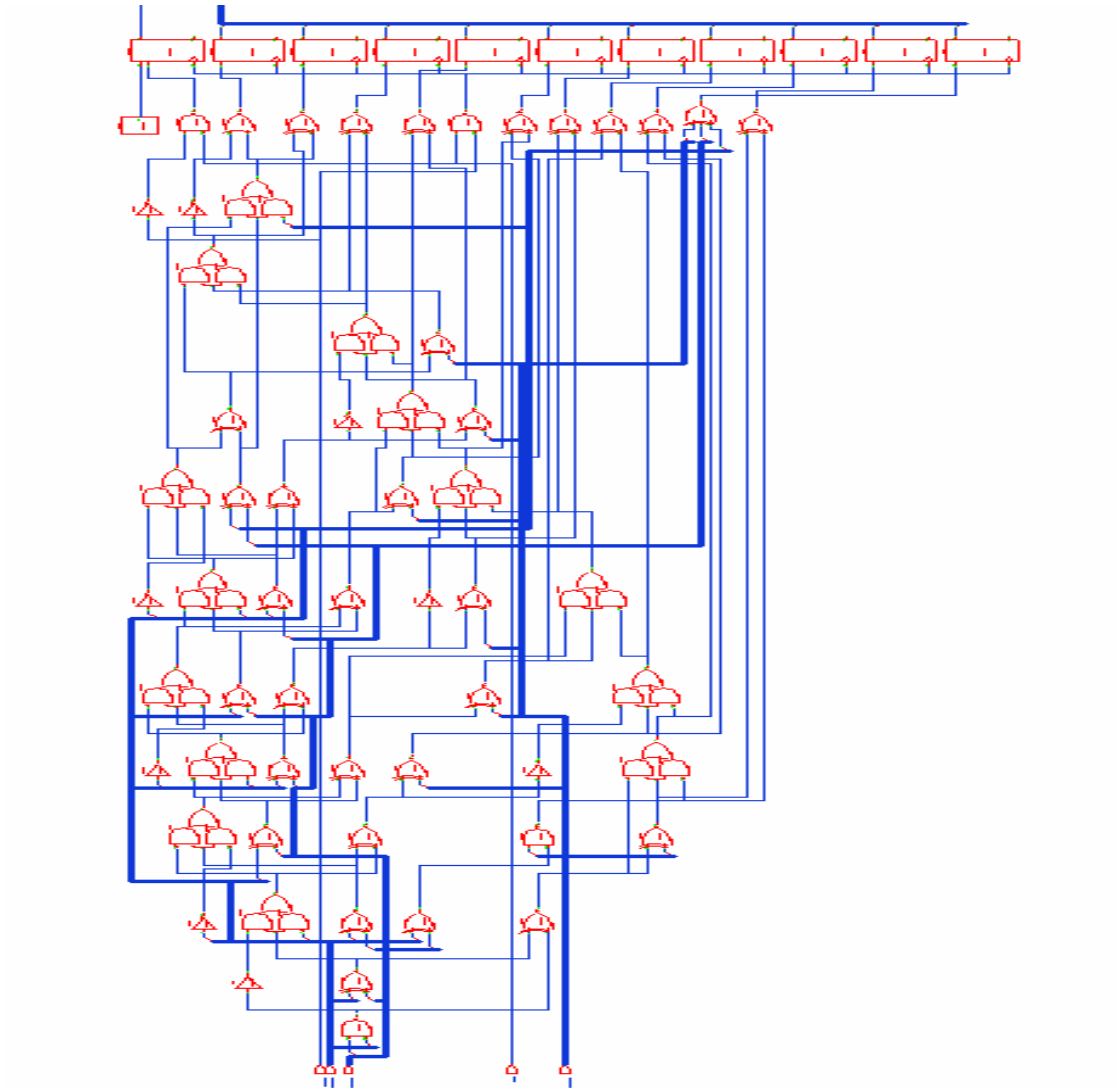


Figure 5-3 ***Layout of Addcolor***

5.3 Divider Unit

This is a very important component of the entire system and also creates a performance constraint. This is because the divide is a cyclic process that is sequential subtraction. That is why then number of clock cycles to perform this depend up the value of number to be divided. This unit always does divide by three to complete the operation of gray scale conversion. It takes the input from the previous input and then implements the divide operations. There is other method for this also like shift and add but are complicated. This is the part where further work to improve the system performance can be done. This components have the signals like CLK, START , A , DONE , GRAY. The value comes in A and with START signal high it divides the value in A by 3 and places the value in gray i.e. the equivalent gray color.

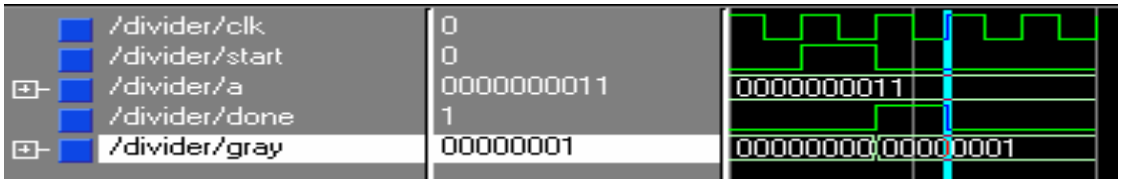


Figure 5-4 ***Timing Diagram***

Below is the partial circuit diagram of divider that has been implemented. The reason it can be called as partial is that in MODEL SIM, the while statements works perfectly while it do not in LEONARDOR SPECTRUM. The other approach to implement will be going through state level architecture changing state on each clock cycle. But it will become an important performance constraint. So a solution has to be evaluated, for it.

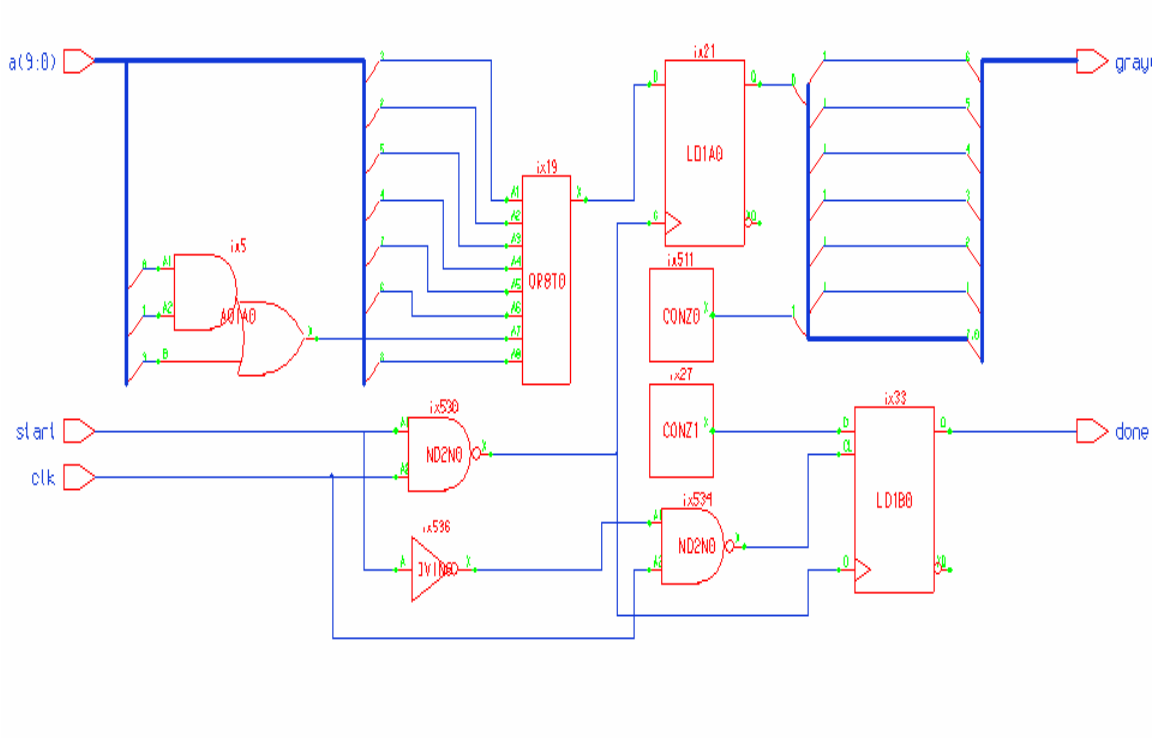


Figure 5-5 ***Layout of Divider Unit***

5.4 Black white converter

In this part of the process we convert the color into black and white. For this system the value of threshold is set at 100 i.e. below this the color is white and after this the color is black. The signals used in this entity are CLK , A, START ,DONE, VAL. These signals are used for both synchronization and exchange of data. After the data is available in A at the CLOCK event when the START is high the conversion is performed and we get the results in next clock cycle.

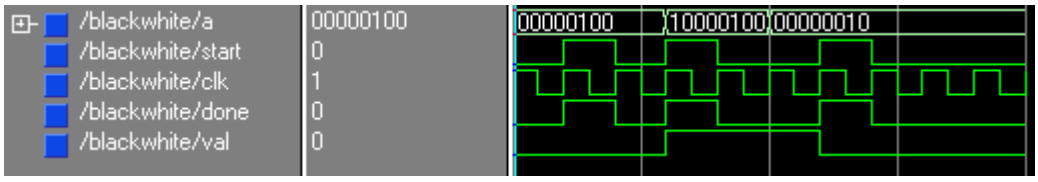


Figure 5-6 ***Timing Diagram***

In the above it can be seen that when START goes high the value of VAL changes accordingly. The component and gate level description of this is given below.

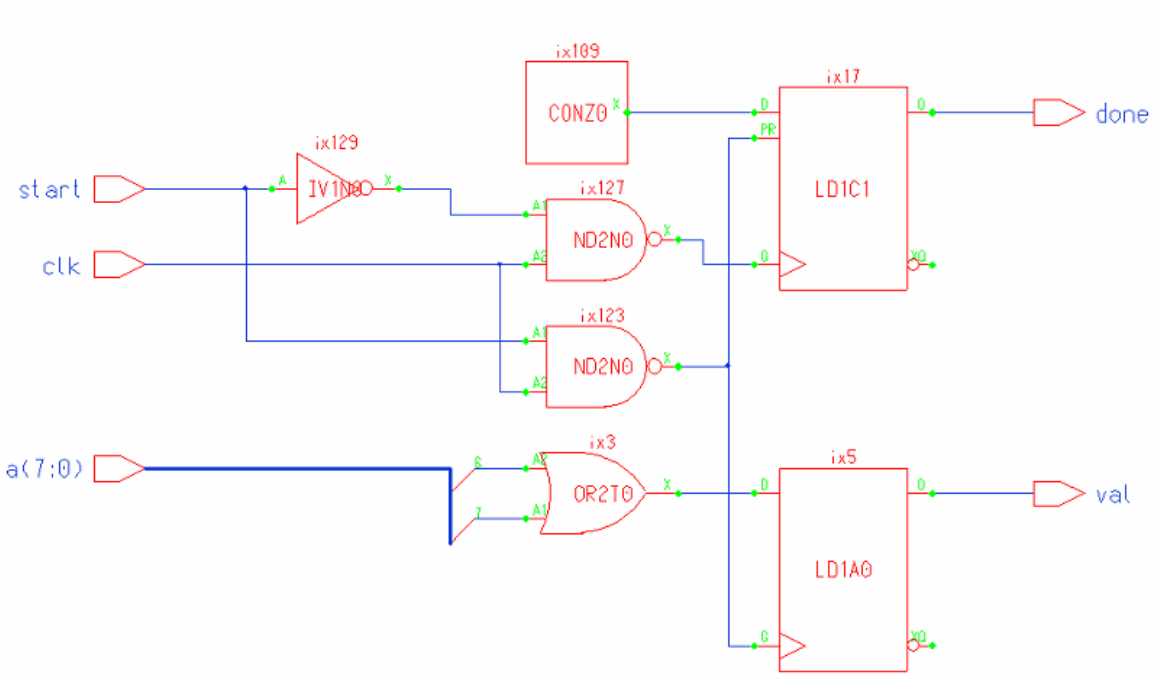


Figure 5-7 ***Black White Converter***

5.5 Count Pixels

This is the calculator of the entire system that is it counts or we can say it adds the number of Black pixels. It contains a register which constantly maintains the record of number of continuous black pixel. When the continuous flow is broken, it tells the control unit that flow was interrupted and then gives the number of black pixels at output. The signals used in this are CLOCK, COLOR, START, COUNT, DONE.

The DONE signal becomes high after the control was broken. And number of continuous pixels are shown in the COUNT signal.

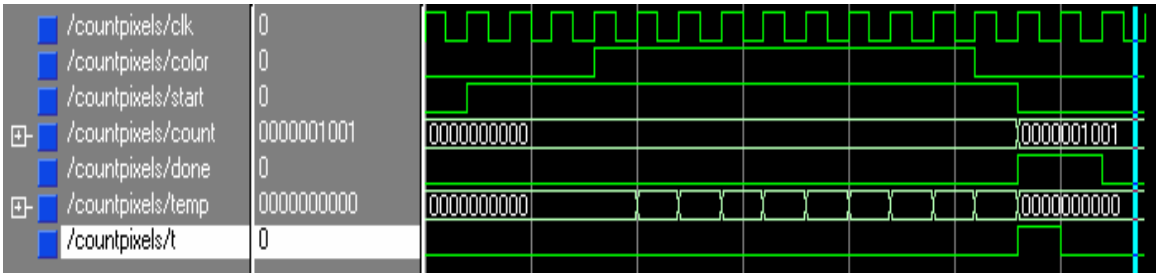


Figure 5-8 T*iming Diagram*

Here we can see that the color bit was high for 9 clock cycle that is there nine continuous black pixels and after that when the white pixel come DONE signal was high and count is available at out put.

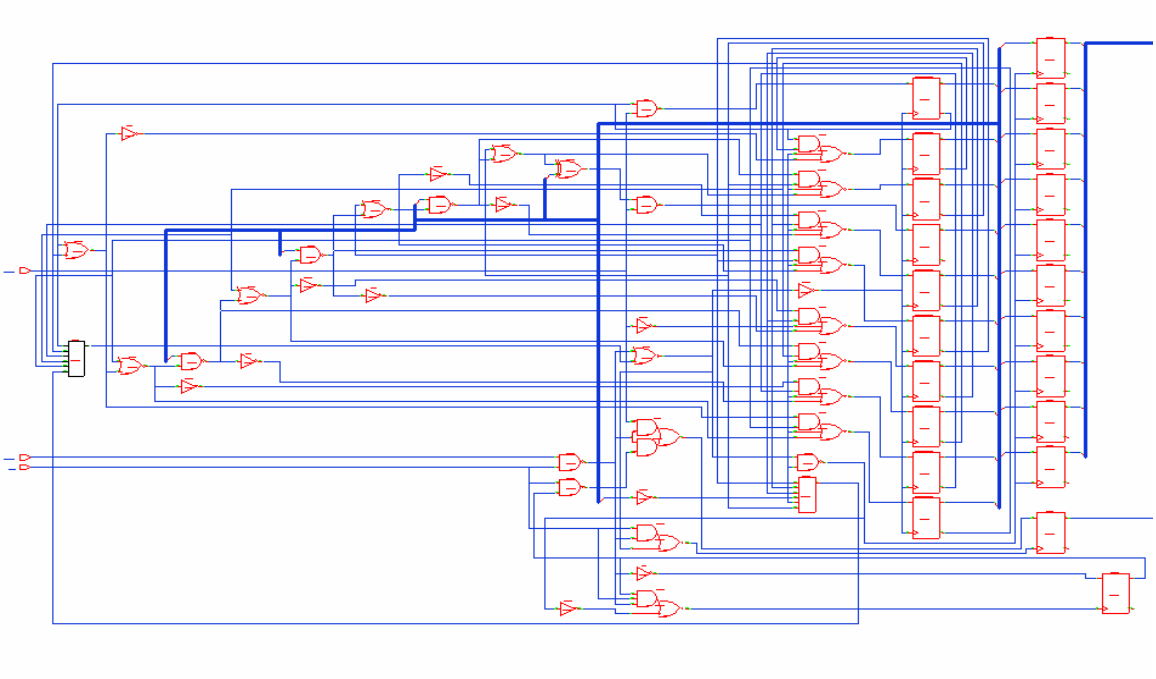


Figure 5-9*Layout Count Pixel*

5.6 Control Unit

Control unit does the usual work of Synchronization between various components of the system. In this circuit it produces the timing signals for call the components i.e. when to start and when the component has done the processing then send the stop signal. The other purpose is to keep the address generation unit intact and ask it to constantly give the address of next location. There are a total of 39 states but broadly there are few states.

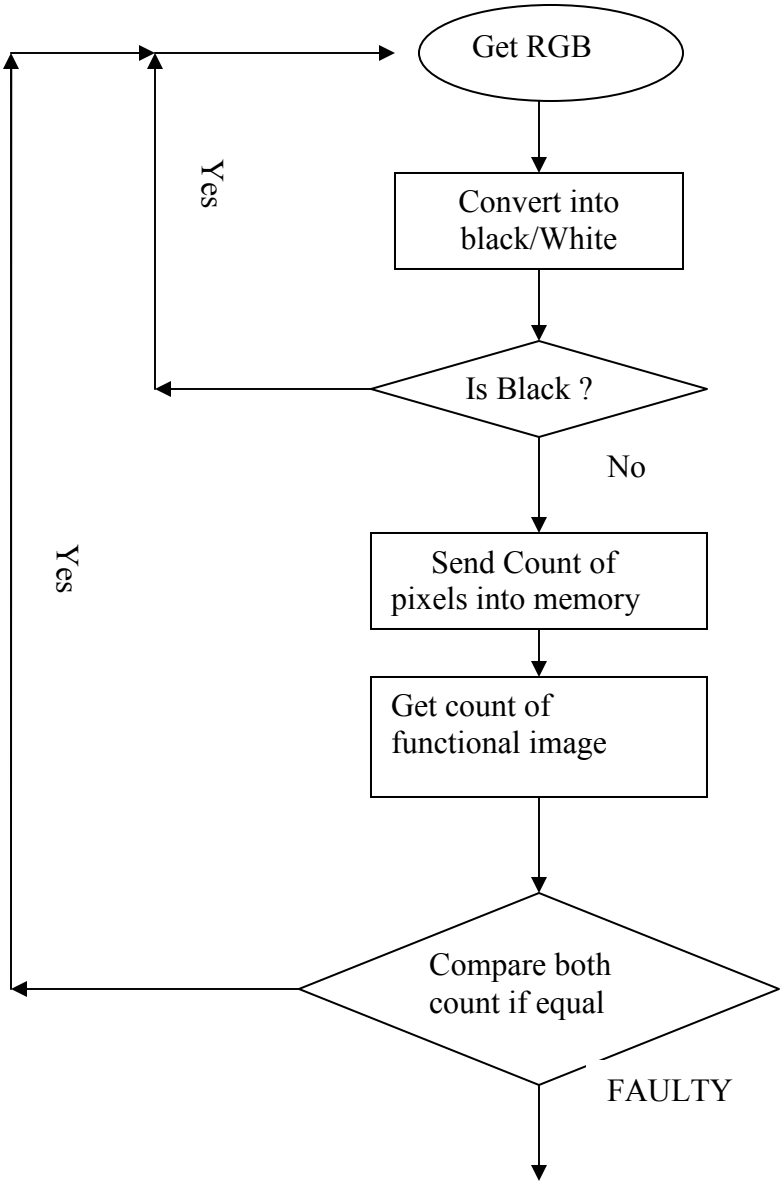


Figure 5-10 ***Flow chart / State diagram***

This is the Approximate State diagram or flow chart followed by the Control Unit. The following is the Complex diagram of control unit and the reason for this is the number of states that has been used in the CU But they can be further minimized to get better control unit performance.

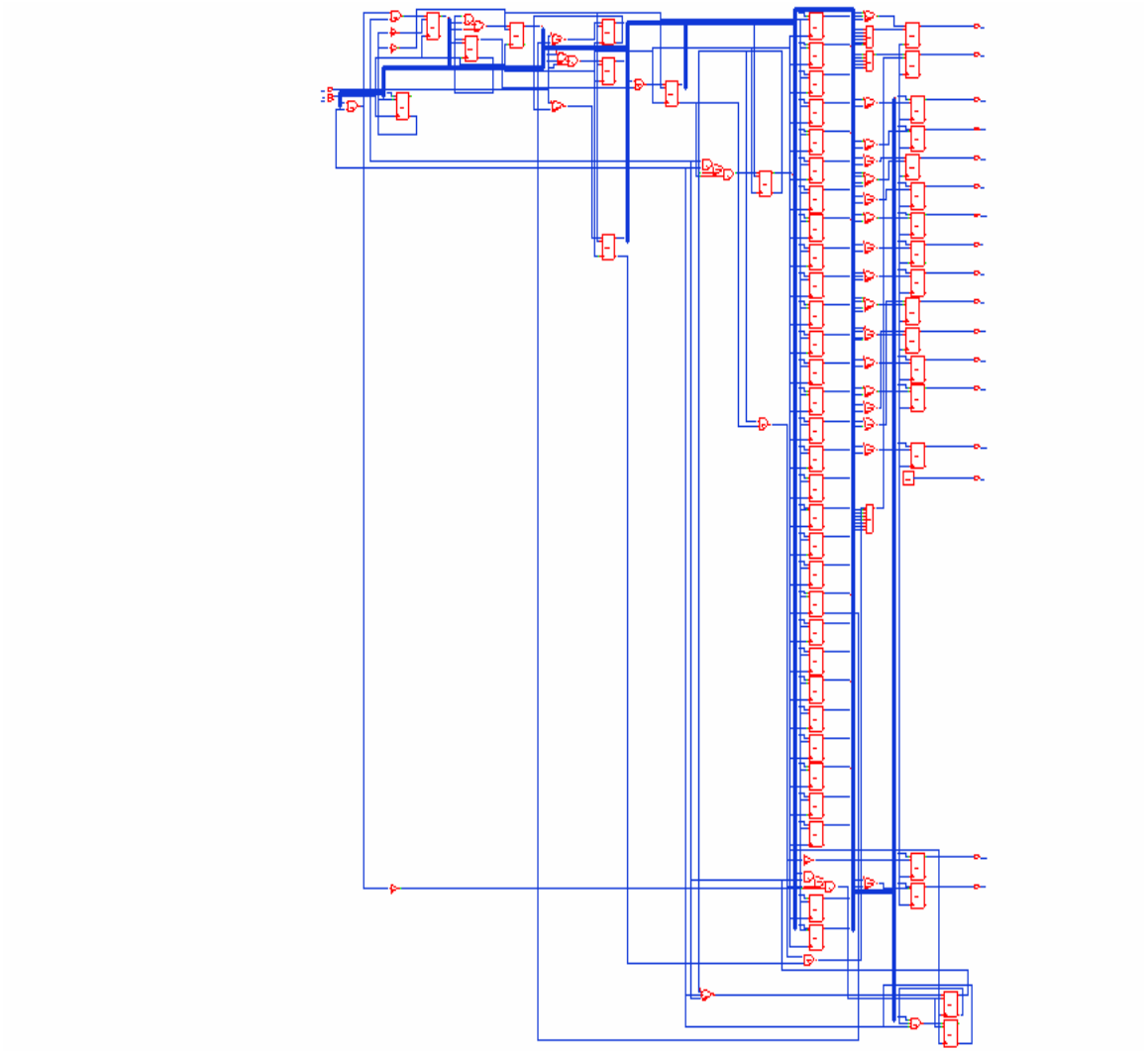


Figure 5-11 ***Layout of Control Unit***

5.7 Address Generation Unit

This unit is responsible for the generation of the address for the System. It gives the next location for where next byte will be picked up. So for a single pixel there are three consecutive memory locations. Every time the CU asks it to generate address it increments one in the previous address. The signals used in this entity are CLOCK, INR, OUT_ADDRESS and WRITE. It can address memory Upto 20 address lines. When clock is high and INR signal is given by the control unit the address generation unit increments the address and makes it available at OUT_ADDRESS.

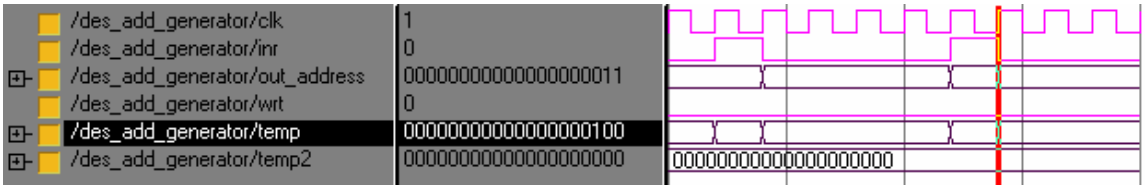


Figure 5-12 ***Timing diagram***

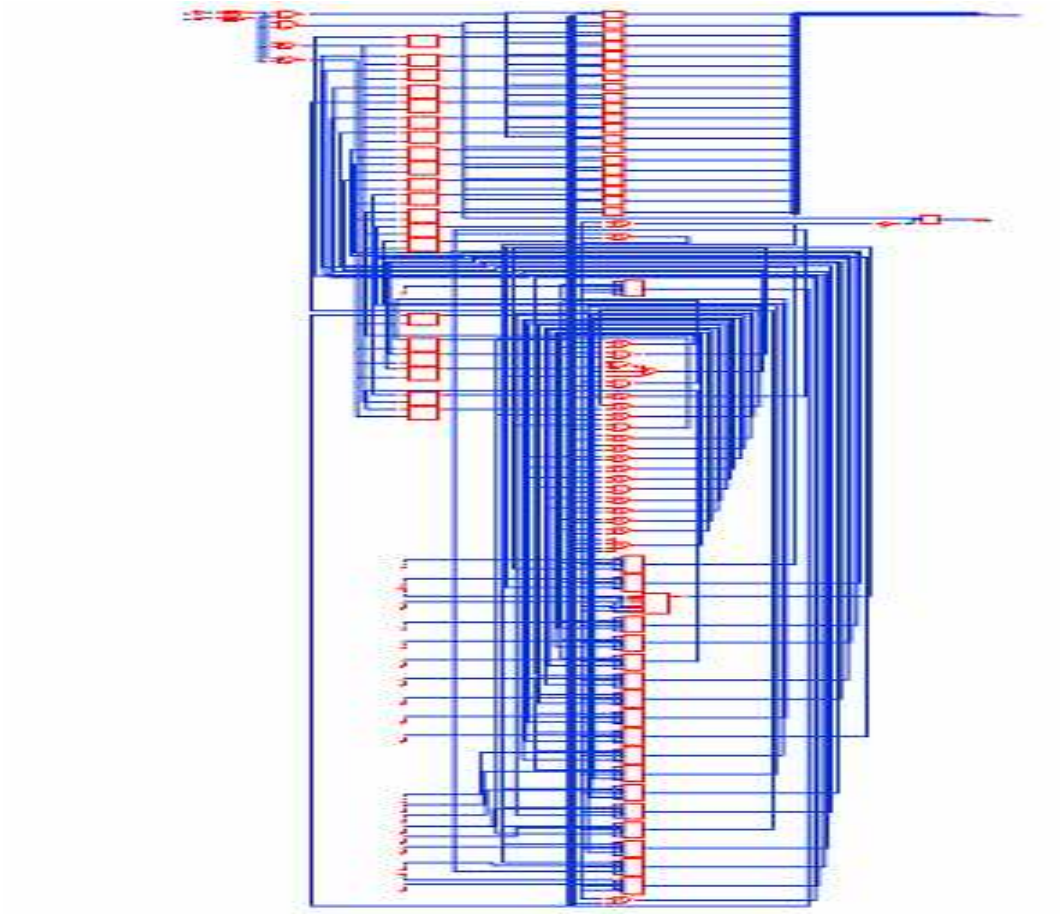


Figure 5-13 ***Layout of Address Generation Unit***

5.8 Write Data

This unit converts the incoming count, which are approximately 10 bytes in length into transferable format. That is first divide the data in to two half's 0 to 7 and rest and then send them to memory unit one by one. The signals are CLOCK LOAD, BIT1, BIT2, VAL, and BYTEVA. First load the value in VAL by LOAD signal and then gain the two bytes by the signals BIT1, BIT2.

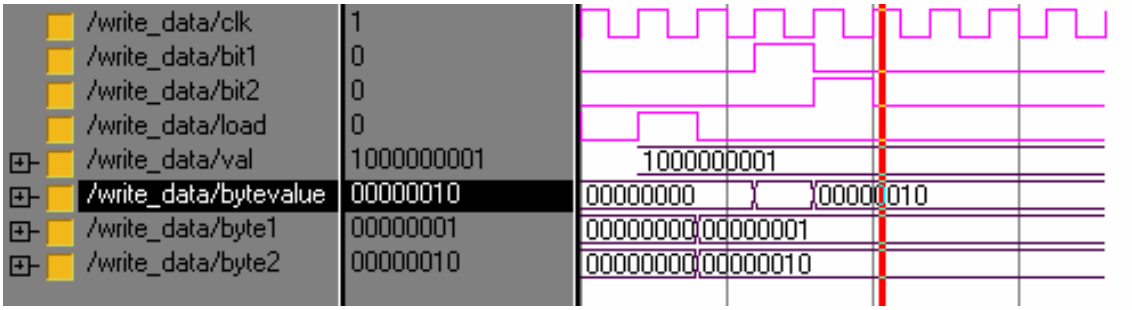


Figure 5-14 ***Timing diagram***

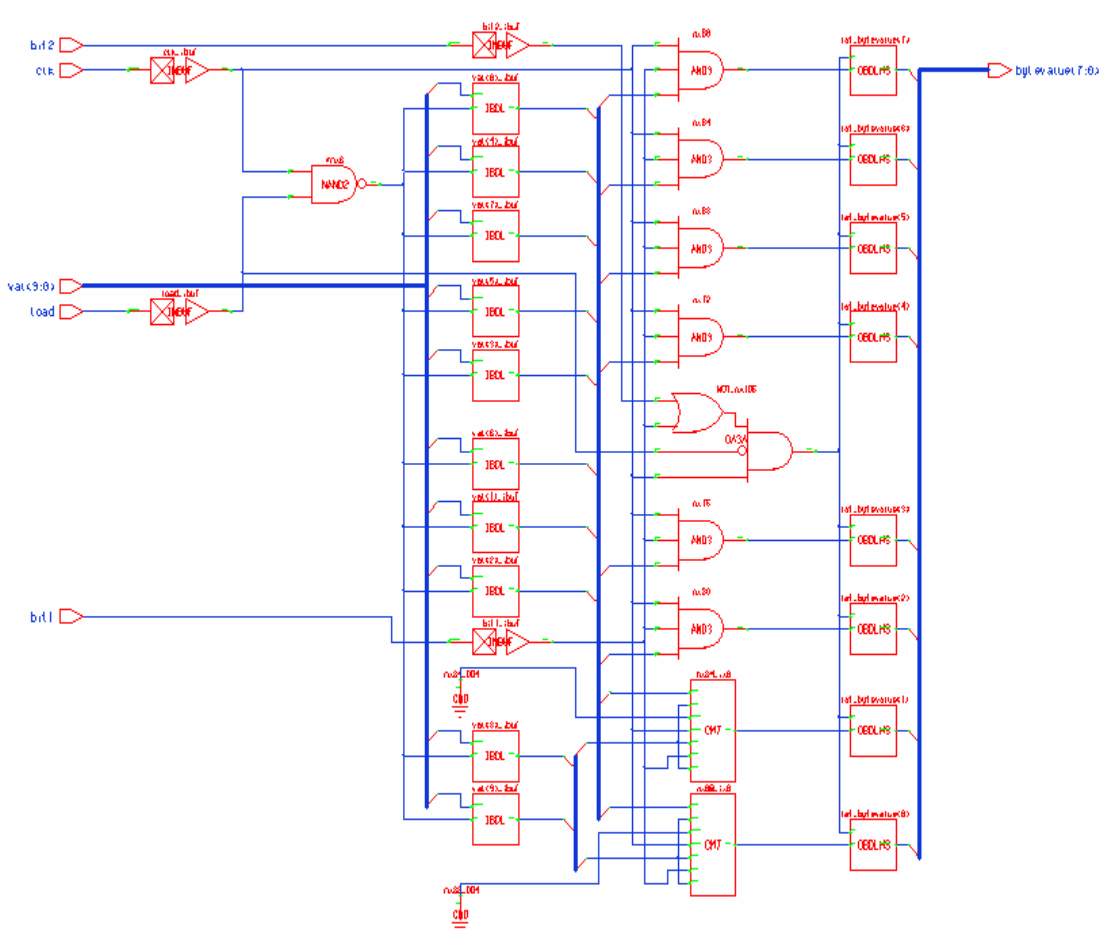


Figure 5-15 ***Write DataUnit***

5.9 Memory Interface Unit:

This unit handles all the interface with the memory that is to send data out chip and bring data into the chip. This simple has ADDRESS and DATA as input then fetches the data from memory. It also has signals like READ, WRITE, CS etc.

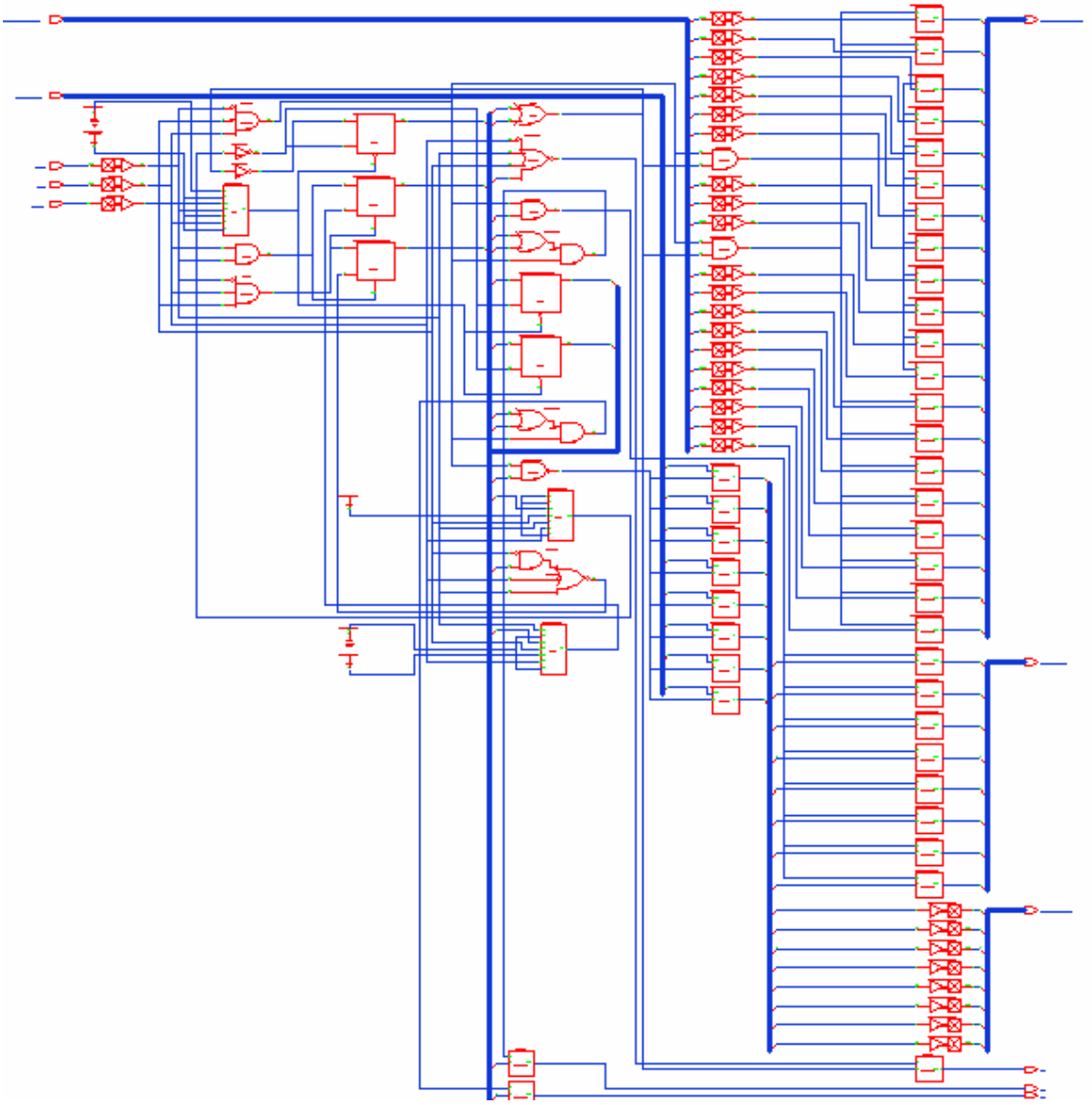


Figure 5-16 ***Memory Interface Unit***

5.10 Comparator

This unit compares the two count and simple tells whether they are equal or not. If not the PCB is faulty. In this the check for equal or not has been implemented, but not for greater or smaller. That is why it is relatively simple to implement.

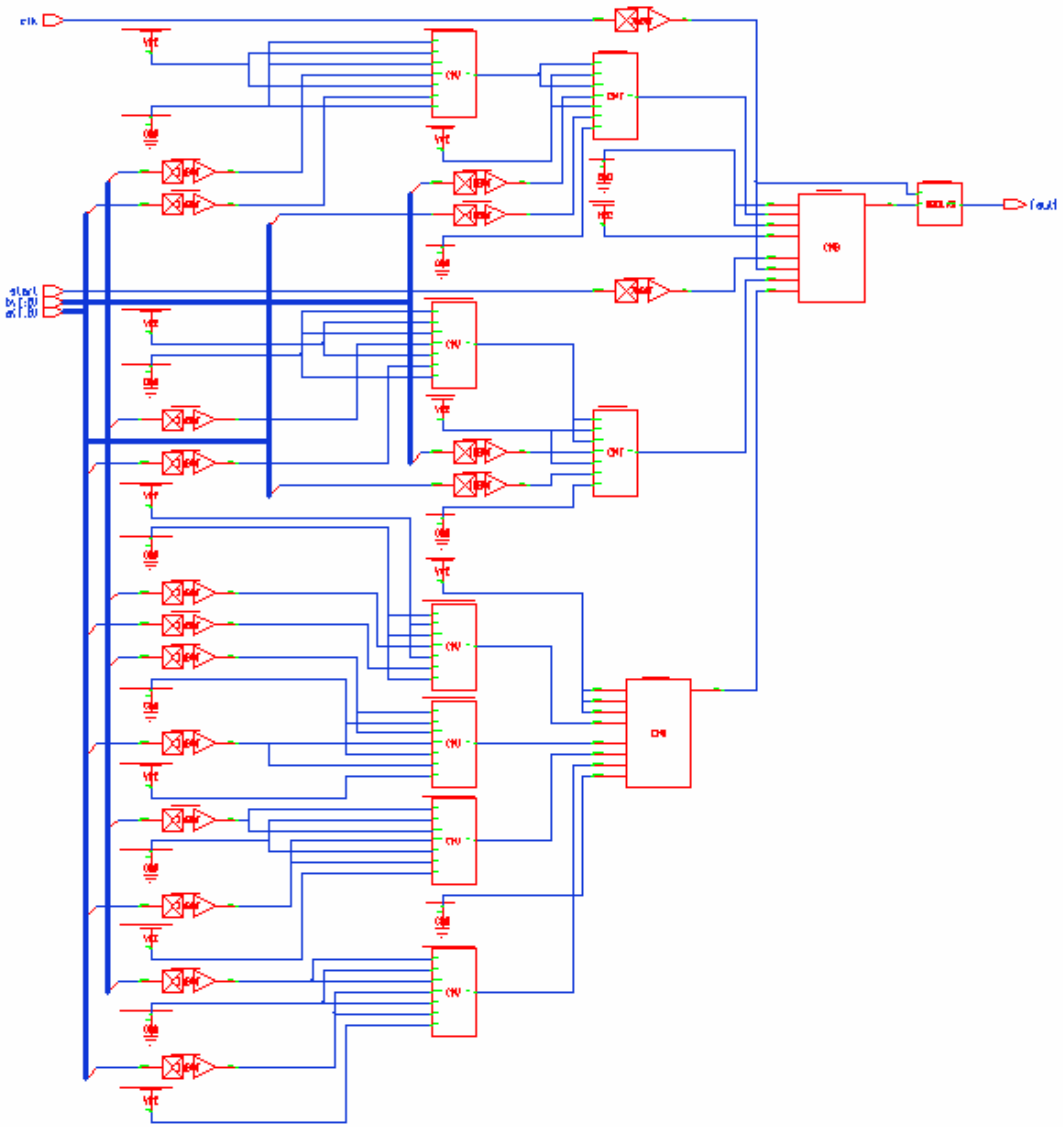


Figure 5-17*Layout of Comparator*

Chapter 6

Results and Discussion

The results have discussed that how much time it took for software to do the tasks like gray scale conversions and runlength encoding and then compared with the results of hardware implementation.

6.1 Software Implementation Results:

Table of timing of various implementations are given below i.e. unsafe code and GetPixel method.

Number of pixels	GetPixel Method Time in mill sec	Unsafe code method Time in mill sec
10000	78	0
40000	140	15
90000	328	47
160000	563	62
250000	890	94
360000	1266	141
490000	1718	203
640000	2016	266
810000	2453	328
1000000	3095	422

Table 6-1 Comparison of GetPixel and Unsafe method

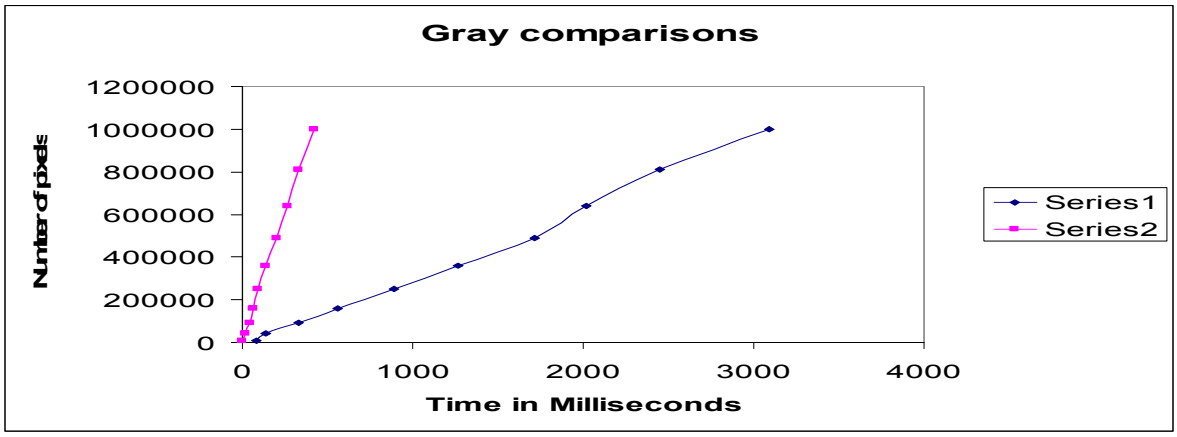


Figure 6-1 ***Graph of comparison between GetPixel and Unsafe method***

The time take for runlength are given below

	Mill. sec.	number of pixels
1	15	10000
2	16	40000
3	31	90000
4	62	160000
5	94	250000
6	140	360000
7	188	490000
8	230	640000
9	312	810000
10	375	1000000

Table 6-2*Time taken to convert Into Runlength Encoding*

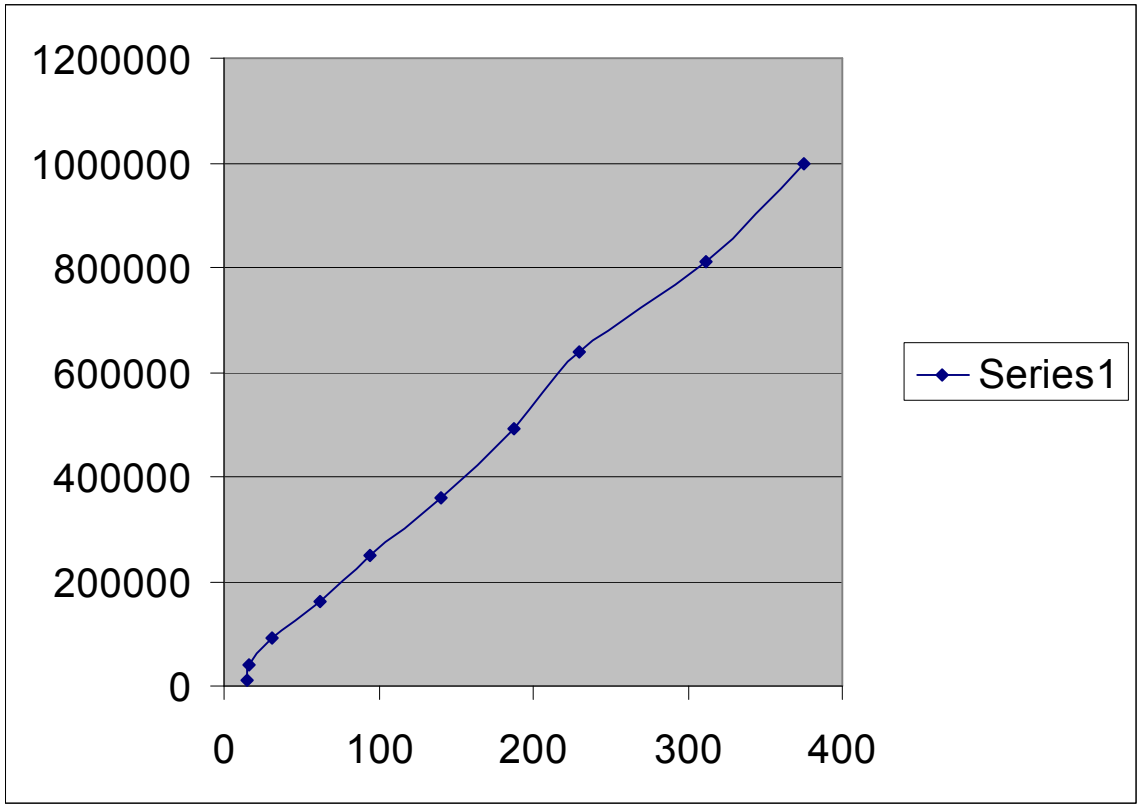


Figure 6-2 ***Graph of time taken for Runlength Encoding***

These all reading are taken on P4 256 MB ram system and software is made in C#. NET and Visual Studio 2003. The test files used are simple (BMP) files of resolution 100*100; 200*200; etc all are red color.

6.2 Hardware Implementations Results:

In the simulation it took 18 clock cycles to process a pixel and 25 CC for comparison so the formula that has been derived to carry out this calculation is as follows.

18*number of pixels * clock time period.

frequency= 1 Mhz.	resolution	Time in mill sec
1	10000	0.01
2	40000	0.04
3	90000	0.09
4	160000	0.16
5	250000	0.25
6	360000	0.36
7	490000	0.49
8	640000	0.64
9	810000	0.81
10	1000000	1

Table 6-3 Time taken in Hardware Implementation

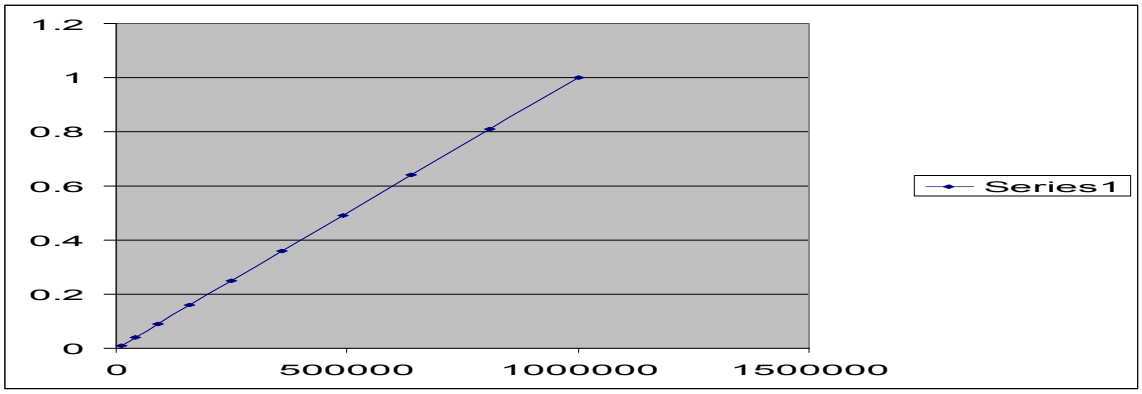


Figure 6-3 ***Graph of time taken for processing in hardware***

Form the results in the previous, it can be seen that the getpixel method is much slow as compared to the unsafe code. After the time for runlength encoding is added, it is found that the time for a file off 1000 * 1000 is app 1 sec. The hardware implementation time of 1000 * 1000 size image on a FPGA working at 1MZ the time is app 1 Sec. So it can be concluded that if the system can be implemented for frequency of more than 1MZ, a performance equal to normal PC can be attained and by achieving the higher frequencies, better results can be obtained. The main limiting factor for this is

the speed of RAM or frequency of RAM and the speed of divider. So better the divider, better the performance.

Chapter 7

Conclusion

The hardware implementations has provided a better perspective as the implementation was confined to dedicated system architecture and could be implemented on FPGA with much better results the software part depends on the Operating system and processor speed which provide many constraints.

The present thesis has derived a new algorithm which is hybrid of runlength encoding and image substitution algorithms and provided a case study with respect to inspection of PCBs

Because of time constraints the present implementation has been confined to the part after the image has been acquired for a perfect implementation of this system, there is a need to embed a camera for image acquisition and dynamic implementation of this system.

REFERENCES

[1] A. Rosenfield, “Machine Vision for Industrv: Concept and Technicrues”, IfI Robotics Today, December 1985.

[2] B. K. Jang and R. T. Chin, "Analysis of Thinning Algorithms Using Mathematical Morphology", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, n. 6,pp. 541-551,1990.

[3] D. Oberkampf, D. F. DeMenthon, and L. S. Davis, ”Iterativepose estimation using coplanar feature points”,Computer Vision and Image Understanding, Vol. 63,
No. 3, pp. 495-511, 1996.

[4] E. J. R. Justino, Metodologia de InspegCO Automdtica de Placas de Circuito Impresso. Curitiba. Centro Federal de Educaqiio Tecnol6gica do Parana - CEFET-PR, 199 1.

[5] J. R. Mandeville, "Novel Method for Analysis of Printed Circuit Boards", IBMJ. Res. Develop., 1985, Vol. 29, pp. 73-86.

[6] J. Mandeville, “Novel Method for Analysis of PCB.” IBM J. Res. Develop. 29 (1985), 73-86.

[7] M. Moganti, F. Ercal, C. Dagli, S. Tsunekawa,“Automatic PCI Inspection Algorithms: A Survey,”Computer Vision and Image Understanding 63(1996), 287-313.

[8] N. Cui, J.J. Weng, and P. Cohem, ”Recursive-batch estimationof motion and structure from monocular imagesequences”, Computer Vision and Image Understanding,Vol. 59, No. 2, pp. 154-170, 1994

[9] Y. Hara, H. Doi, K. Karasaki et al., "A System for PCB Automated Inspection Using fluorescent Light", IEEE Transactionson Pattern Analysis and Machine Intelligence, Vol. 10, n. 1, pp. 69-78, 1988.

[10] Y. Sun, C Tsai. “A New Model-Based Approach for Industrial Visual Inspection.” Pattern Recognition25 (1992), 1327-1336.

[11] Gram Smith “Complete Reference to C#.NET “

[12] Willson b Sethy “C# Exposed”

[13] WWW.Microsoft.Com

[14] WWW.MSDN.COM

Publications

[1] Anupam Sharma, Navjot Kaur, Pravesh Sani, “FPGA based vision system for PCB inspection”, WNES 2006 Acceptance.

[2] Anupam Sharma, Navjot kaur, Kulvinder S Mann, “Review of Implementation of Neural Network in VLSI ”, ECCS 2006.

[3] Raghu, Navjot Kaur, Anupam Sharma “Design of Real Time Schedulers for Embedded System using Hardware/Software Co design Appoach” ECCS 2006.