SIMULATION OF FPGA-BASED IMAGE PROCESSING SYSTEM FOR QUALITY CONTROL AND PALLETIZATION APPLICATIONS

by

Abubakar Muhammad Ashir

Submitted to the Institute of Graduate Studies in Science and Engineering

in partial fulfillment of

the requirements for the degree of

Master of Science

in

Electrical-Electronic Engineering

Mevlana (Rumi) University

2014

SIMULATION OF FPGA-BASED IMAGE PROCESSING SYSTEM FOR QUALITY CONTROL AND PALLETIZATION APPLICATIONS

submitted by **Abubakar Muhammad Ashir** in partial fulfillment of the requirements for the degree of Master of Science in Electrical-Electronic Engineering Department, Mevlana (Rumi) University

APPROVED BY:

Examining Committee Members:

Assist. Prof. Dr. Mohammad Shukri SALMAN

(Thesis Supervisor) . . . . . .

Prof. Dr. Atef A. ATA

(Thesis Co-Supervisor) . . . . . .

Assoc. Prof. Dr. Nihat YILMAZ . . . . . .

Assoc. Prof. Dr. Essam ABO SERIE . . . . . .

Assist. Prof. Dr. Alaa ELEYAN . . . . . .

Prof. Dr. M. Uğur Ünver . . . . . .

Head of Department, Electrical-Electronic Engineering

Assoc. Prof. Dr. Ali Sebetci . . . . . .

Director, Institute of Graduate Studies in Science and Engineering

DATE OF APPROVAL (/ 06 / 2014)

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Abubakar Muhammad ASHIR

Signature:

ABSTRACT

SIMULATION OF FPGA-BASED IMAGE PROCESSING SYSTEM FOR QUALITY CONTROL AND PALLETIZATION APPLICATIONS

MASTERS OF SCIENCE THESIS, 2014

Thesis superviser: Assist. Prof. Dr. Mohammad Shukri Salman

Thesis co-supervisor: Prof. Dr. Atef A. ATA

Keywords: FPGA, Belt Conveyor, Simulink, Quality Control, Edge Detection.

This thesis proposes an approach for solving famous industrial application problems: Quality Control and Palletization (QCP). An intelligent four-bar mechanism has been designed as mechanical palletizer. It was modelled as a singular quadralateral mechanism whose intelligence is sourced from an image processing algorithms. The algorithms are targeted for Field Programmable Gate Array (FPGA) processor to improve processing speed and meet requirements for real-time processing systems. The generic techniques for handling palletization application problems in the industries are based on sensor-based mechanical device and robot. While sensor-based mechanical palletizers are known for low efficiency and inability to handle complex tasks, robot-based palletizers are expensive and experience delay in real-time applications due to the fact that the image processing algorithms are executed in General Purpose Processor (GPP).

In this approach, an intelligence-sourcing mechanical palletizer and and FPGA architecture are proposed to handle QCP applications. The algorithms are implemented using MATLAB and Simulink packages. The critical system blocks of the Simulink model are the serial pixel data generator and the thresholder whose functions is to compute threshold value of all pixels for binarization. All the Simulink system blocks have been designed based on the proposed

FPGA architecture and mapped onto the Configurable Logic Blocks (CLB) of the FPGA. The Hardware Description Language (HDL) codes generated from the Simulink model show no behavioral deviation from the original MATLAB version of the algorithm. The recognition rate results are high and the whole system is very fast at 50 MHz clock frequency.

ÖZET

KALITE KONTROL VE PALETLEME UYGULAMALARI IÇIN FPGA TABANLI GÖRÜNTÜ IŞLEME SISTEMI SIMÜLASYONU

Yüksek Lisans Tezi, 2014

Tez Danışmanı: Yrd. Doç. Dr. Mohammad Shukri Salman

Tez Eş-danışmanı: Prof. Dr. Atef A. ATA

Anahtar Kelimeler: FPGA, Konveyör Bant, Simulink, Kalite Kontrol, Kenar Algılama.

Kalite Kontrol ve Paletleme (QCP): Bu tez, ünlü endüstriyel uygulama sorunlarını çözmek için bir yaklaşım önermektedir. Bir akıllı dört-çubuk mekanizması mekanik paletleyici olarak dizayn edilmiştir. Bu kimin istihbarat bir görüntü işleme algoritmaları kaynaklı bir tekil dörtgen mekanizma olarak modellenmiştir. Algoritmalar Alan Programlanabilir Kapı Dizisi işlem hızını artırmak ve gerçek zamanlı işleme sistemleri için gereksinimleri karşılamak için (FPGA) işlemci için hedeflenir. Sektörlerde paletleme uygulama sorunları ele için genel teknikler sensör tabanlı mekanik cihaz ve robot dayanmaktadır. Sensör tabanlı mekanik paletleme düşük verimlilik ve karmaşık görevleri ele yetersizlik için bilinmesine rağmen, robot tabanlı paletleyicilergörüntü işleme algoritmaları (GİH) Genel Amaçlı İşlemci yürütülür olması nedeniyle gerçek zamanlı uygulamalar pahalı ve deneyim gecikme vardır.

Bu yaklaşımda, bir istihbarat-kaynak mekanik paletleme ve FPGA mimarisi QCP uygulamaları işlemek için önerilmiştir. Algoritmalar MATLAB ve Simulink paketleri kullanılarak uygulanır. Simulink modelinin kritik sistem blokları seri piksel veri jeneratör olan ve fonksiyonları Binarization için tüm piksel eşik değerini hesaplamak için eşileleyici vardır. TümSimulink sistem bloklarıönerilen FPGA mimarisine dayalı tasarlanmış ve FPGAyapılandırılabilir lojik blok (CLB) üzerine haritalanmıştır. Donanım Tanımlama Dili (HDL) Simulink model oluşturulur kodları algoritmanın orijinal MATLAB sürümünden

hiçbir davranışsal sapma gösteriyor. Tanıma oranı sonuçları yüksek olan vetüm sistem 50 MHz saat frekansında çok hızlı.

To my mother: Hafsat Ahmad Rufa’i.

&

My wife: Faiza Abubakar Ashir.

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my supervisor Assist. Prof. Dr. Mohammad Shukri SALMAN and co-supervisor Prof. Dr. Atef A. ATA for their guidance, advice, criticism, encouragement and insight throughout the research.

My appreciations and heartfelt gratitudes go out to all who particularly contributed in one way or another towards my success in life, and in the course of this thesis, in particular. Worthy to mention are my parents, immediate family, friends, relatives (both close and distance) and associates. Your advices, goodwill messages and prayers would be remembered for always. I love you all.

I am particularly indebted to his Excellency, the revered peoples' governor, Dr. Rabiu Musa Kwankwaso for his foresight in piloting the scholarship programme which provided us with opportunities to attain this feat. You will forever continue to endear in the bossom of our hearts for your unparallel contributions in our state and beyond.

TABLE OF CONTENTS

ABSTRACT iv

ÖZET vi

ACKNOWLEDGEMENTS ix

TABLE OF CONTENTS x

LIST OF TABLES xii

LIST OF SYMBOLS/ABBREVIATIONS xv

CHAPTER ONE 1

1. INTRODUCTION AND OBJECTIVES 1

1.1. Introduction 1

1.2. Relevent Literature Review 2

CHAPTER TWO 11

2. PROBLEM STATEMENT 11

2.1. Formulation of the Problem 11

2.2. Objectives of the Thesis 15

CHAPTER THREE 16

3. CONCEPTUAL DESIGN AND MATHEMATICAL MODELLING 16

3.1. Image Processing Algorithms 16

3.1.1. Morphological Operation 16

3.1.2. Connected Components Analysis 18

3.1.2.1. Geometrical Features Extraction 19

3.1.3. Object Recognition 20

3.2. FPGA Architectural Design and Implementation 22

3.2.1 FPGA Design Architecture 22

3.2.2 FPGA Implementation of the Algorithm 23

3.2.3 Data Serialization 25

3.2.4. CCD Camera and the FPGA Speed Analysis 26
3.3. Mechanical System Modelling 27
3.3.1. Mathematical Model of the Belt Conveyor 27
3.3.2. Modelling of the Orientation Mechanism 31
3.3.2.1. Displacement and Velocity Computation 31
3.3.2.2. Linkage Synthesis and Kinematics at Constant Velocity 33
3.3. Synchronization and Timing of the Multi-body System 33
CHAPTER FOUR 36
4. RESULTS AND DISCUSSIONS 36
4.1. Introduction 36
4.1. Control Flow 36
4.2. Image Processing Simulation Results 39
4.4. Belt Conveyor Simulation Results 43
4.5. Orientation Mechanism Simulation Results 47
4.6 Discussions 49
CHAPTER FIVE 50
5.1. Conclusions 50
5.2. Future Work 50
APPENDIX A 51
REFERENECES 52

LIST OF TABLES

Table 4.1	Summary Experimental Result	40
Table 4.2	Induction Motor Parameters	44

LIST OF FIGURES

Figure 3.1: 4-neighborhood connectivity operator 19

Figure 3.2: 8-neighborhood connectivity operator 19

Figure 3.3: BLOBs with an ellipse of same second moment 20

Figure 3.4: FPGA chip 22

Figure 3.5: FPGA architecture 23

Figure 3.6: 4×4 Pixel generation architecture 24

Figure 3.7: Implementation stages 26

Figure 3.8: Schematic diagram of the belt conveyor with rotor circuits 28

Figure 3.9: Stator equivalent circuit diagram 28

Figure 3.10: Rotor equivalent circuit diagram 29

Figure 3.11: Schematic diagram of a four-bar mechanism 33

Figure 3.12: Schematic diagram of the objects time estimation on the conveyor 34

Figure 4.1: Proposed system diagram 37

Figure 4.2: Control flow chart 38

Figure 4.3: Simulink model 39

Figure 4.4: Template 1 41

Figure 4.5: Template 2 41

Figure 4.6: Template 3 41

Figure 4.7: Template 4 41

Figure 4.8: Template 5 41

Figure 4.9: Template 6 41

Figure 4.10: Template 7 41

Figure 4.11: Template 8 41

Figure 4.12: Template 9 41

Figure 4.13: Template 10 42

Figure 4.14: Template 11 42

Figure 4.15: Template 1 pre-and post-processing results 42

Figure 4.16: Template 2 pre-and post processing results 42

Figure 4.17: Template 3 pre-and post processing results 42

Figure 4.18: Template 4 pre-and post processing results 43

Figure 4.19: Pre and post-processing results of unmarked template 43

Figure 4.20: Post- processing Results of Non-target Image 43

Figure 4.21: Belt conveyor Simulink model 45

Figure 4.22: Belt conveyor speed vs load 46

Figure 4.23: Belt conveyor tension vs load 46

Figure 4.24: Plot of Trajectory of joint B against time 48

Figure 4.25: Plot of Trajectory of joint C against time 48

LIST OF SYMBOLS/ABBREVIATIONS

Symbol	Explanation
μ_k	Kinetic Friction Coefficient
ω	Angular Velocity
T_0	Output Torque
ANN	Artificial Neural Network
BLOB	Binary Large Object
CCD	Couple Charged Device
CLB	Configurable Logic Blocks
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
GPP	General Purpose Processor
HDL	Hardware Description Language
HOG	Histogram of Oriented Gradient
LBP	Local Binary Pattern
MLP	Multi-Layer Perception
OD	Orientation Device
RTOS	Real-time Operating System
OCR	Optical Character Recogniton
PLC	Programmable Logic Controller
SE	Structural Element
SIFT	Scale Invariant Feature Transform

CHAPTER ONE

1. INTRODUCTION AND OBJECTIVES

1.1. Introduction

Following compelling advancements in sensors and digital tecnology, industries have been shifting their focus towards full automation. Processes like Quality Control and Palletization (QCP) are among the most recurring routines in the manufacturing industries. As the manufacturing processes make transition towards full automation, numerous approaches have been developed and adopted using different techniques for optimal performance and accuracy.

Machine vision has been one of the hot research topics that have received high level of attention in industrial applications. Output of the vision analysis provides platforms for decision making and contol signals that currently make automation a much more realistic process. The vision system in turn consists of huge calculations and processing of the images of a process captured by high resolution industrial Charged Couple Device (CCD) image sensors [1]. A great deal of processing speed is required to accompish a given task. The need becomes even much high when the application is performed in real-time where the stream of image frames must be processed by the hardware and the result is passed to the next process. Traditionally, General Purpose Processor (GPP) based processing have been the norms for years, however with high demands in speed to process colored images, GPP-based processing grapples with so many challenges [2]. Field Programmable Gate Array (FPGA) based image processing has shown greater processing speed and flexibilty, especially, with recent developments in the FPGA hardwares, making it to be a favorite for real-time processing [3-5].

With the high speed handling capability of the FPGA, it is more convinient and efficient to solve QCP processes with this processor. Quality control processes require vital features extraction from the images to seperate good productions from the defective ones. Meanwhile, palletization processes are only concerned with the geometrical features of the product such as its global coordinate system and orientation. The two applications can be implemented together on the FPGA with true parallel processing.

1.2. Relevent Literature Review

GPP-based image processing algorithms are coded in higher level languages like C and C++ then compiled in vision softwares (e.g., OPENCV, MATROX, SOPOST, etc). The codes execution time and the sensor quality further delay the processing speed of the entire process. The sensor quality is defined by the characteristics such as resolution, noise and timing parameter such as update rate, latency, update rate deviation (jitter) and the algorithms runtime [6, 7]. While some of the constraints can be addressed by the use of high resolution CCD cameras, most of the timing constraints will have to be confronted by the GPP processor and this has several drawbacks [1]. For vision-based sensor systems, the update rate is comparatively low because a full image has to be acquired and transferred with up-date rates of 15 to 30 Hz for USB- or FireWire-cameras [8-10].

Jitters which can be identified as time variation in a periodic signal (e.g. update rate), are major problems in software-based image processing on GPP because of the unpredictable scheduling of the operating system [9]. The latency of the camera-based sensors is usually high, at least one update interval because an object position is calculated after a full image was captured by the camera. All the timing challenges can be handled efficiently using hardware-based image processing deploying FPGA with configurable hardware.

Xilinx FPGA with Digital Signal Processors (DSPs) has been demonstrated to have great ability to accelerate the algorithms by the parallel computation and pipelined structure. They can start the image processing on partial data before the full image is available in the memory [11, 12]. Wang *et al*. [12] demonstrated the parallelism of FPGA using two functional blocks. The first block is used to obtain the threshold value for the image frame and other block is applying the threshold value to the frame. This parallelism and the simple hardware component of both blocks make this approach suitable for real-time applications. The performance remains comparable to the some technique frequently used in off-line threshold determination [13].

To implement the image recognition algorithms, models of the object to be identified are established in the system database. These algorithms generally perform the task of image segmentation, detection, recognition and tracking (e.g. in visual servoing) [6, 14]. Image segmentation is the first step leading to image analysis and interpretation. The goal is to separate the image into regions that are meaningful for the specific task. Segmentation techniques utilized, can be classified into one of five groups [11, 14]: threshold-based, edge-

based, region-based, classification (or clustering)-based and deformable model-based. Each of these methods adopted different techniques to achieve the set of goals and sometimes can be used for specific application. Based on the segmented image the object recognition has two perspectives which include; utilizing appearance features of the objects such as the color and intensity (gray scale) level, or matches the features of interest extracted from the image to the feature stored in the databse. The feature-based approaches have the ability to recognise objects in the presence of poor lighting, translation, rotation and scale changes [13]. To detect these feature some major detectors like Harris detector, Local Binary Partern (LBP) and Histogram of Oriented Gradients (HOG) algorithms are known to be used. The detectors are not the final stage, some sets of trained instructions known as clasifiers are used for comparing the detected object with its models for verification and matching. For a less complicated task, rule-based classifiers are known to be effective e.g., cascaded detectors using Viola-Jones detection algorithm [15, 16]. As the task becomes more complicated artificial intelligence based classifiers has been implemented and have been very effective e.g Artificial Neural Network (ANN) classifier using Multi Layer Perception (MLP) algorithms [16].

One of the most critical parts of infomation extraction from the stream of images frames is the edge detection. From the detected edges vital information such as the geometry, area, centroids, bounding box of an object are revealed using techniques such as the Connected Components Analysis (CCA) of a Binary Large Objects (BLOBS). Edge detection is identified as one of the basic characteristics of the image. It is a useful platform that becomes a basis in the field of image analysis such as; area identification and extraction, image segmentation and object recognition and other regional forms [11]. It is widely used in image segmentation, image recognition, and texture analysis. In addition, Edge detection algorithms are not only limited to detecting the image gray value of a pixel discontinuity, but also determine the exact locations of such pixels [11]. Luckily enough, in FPGA hardware resourses there are sufficient internal multipliers which support complex edge detection requirements such as; the Gaussian noise removal, and so on. The design process can directly call these resources to operate, so it is easy to implement complex convolution. Basically edge of an image has two properties which include; the direction and magnitude.

Usually, the change of the gray value of an image pixel along the edge is uniform, but the pixels perpendicular to the edge have their gray values change dramatically. According to the properties of pixel intensity change, it can be divided into step-type and roof type [17]. In the

step type, both sides of the pixel in value change significantly, and in roof type, it is located in the gray scale to reduce the rate of change from the turning point. The edge of the image shows discontinuous change in gray scale. There is a remarkable difference when comparing the object edge pixels and the adjacent pixels in the gray level region. This is according to the normal basis for the edge detection algorithm. In Classical edge detection algorithms, edges are computed based on the image of a certain neighborhood of each pixel gray value of transformation, using the edge of near direction leads to the first and second derivatives of several changes [4, 17]. For digital images, gray value of image intensity of a significant change in gradient can be expressed by the derivation of the image function gradient. Edge information can be obtained and image can be processed. The traditional edge detection operators are Canny, Roberts's operator, Sobel operator, Laplacian operator, Kirsh operator, Prewitt operator, etc. [4]. From these, Roberts, Sobel and Prewitt operators are the first order derivative-based edge detection operators. Such operators detect image edge by calculating the gradient of image pixels. Usually, near the region of edge of an image region a wide response is obtained, and so wide edge can be detected. Laplacian, LOG and Canny operators are second order derivative based edge detection operators. Such operators detect the edge by calculating the second derivative of the zero-crossing. Because this derivative tests the width of the edge to be thinner, so the precise location of the edge will be easily obtained [4, 14, 18].

Canny has proven to be superior over many of the available edge detectors hence it is prefered for real-time applications [19]. Canny detector, basically, applies four computational steps to detect edge pixels. In the first step, a Gausian mask is applied for image smoothening and noise reduction, and then it computes the first order derivatives for all pixel locations (x, y) in both horizontal and vertical directions. The third phase, applies hysteresis thresholding based on two set threshold values. All pixels with magnitudes greater than the upper threshold are set as edges and those with values between the thresholds and adjacent to an edge pixel are set as edges if both pixels direction are within $\pm 12.5\%$ of each other. Finally, non-maximal suppression is applied to eliminate pixels on edges that are not maxima [14, 19].

The Sobel operator which is a first order classical edge detector has a diffrent approach than Canny dectector. Technically, the algorithm performs a discrete differentiation operation computing an approximation of the gradient of the image intensity function. At each point within the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector [3, 4]. The Sobel operator is based on convolving the image with a small, separable and integer valued filter in the horizontal and vertical directions and is

therefore, is relatively inexpensive in terms of computations. On the other hand, the gradient approximation that produces is relatively vague, in particular, for high frequency variations in the image [17].

The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives. The approximations of the derivatives are computed in both horizontal and vertical directions. The two convolution kernels, each for vertical and horizontal derivatives, are 2-dimensional and hence can be decomposed as the products of an averaging and a differentiation kernel. For this reason, the gradient is computed with smoothing [6]. For example, the *x*-coordinate is defined as increasing in the "right"-direction, and the *y*-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude by summing the absolute values of both Kernels. For digital images, the intensity function of that image is only known at discrete points. Derivatives of this function cannot be defined unless it is assumed that there is an underlying continuous intensity function which has been sampled at the image points [6, 12]. With some additional assumptions, the derivative of the continuous intensity function can be computed as a function on the sampled intensity function, i.e., the digital image. It turns out that the derivatives at any particular point are functions of the intensity values at virtually all image points. However, approximations of these derivative functions can be defined at lesser or larger degrees of accuracy.

The Sobel operator represents an inaccurate approximation of the image gradient, but it is still of sufficient quality to be practically used in many applications [17]. More precisely, it uses intensity values only in 3×3 regions around each image point to approximate the corresponding image gradient, and it uses only integer values for the coefficients which weight the image intensities to produce the gradient approximation.

On the other hand, edge drawing techniques has completely different approach to classical edge detectors. Classical edge detection algorithms work blindly by applying different sets of filters and thresholding, which result in edgemaps that consist of individual edge pixels with no real relationship and connectivity [18]. That is, the edge pixels in the resulting edgemaps are not analyzed for thinness and connectivity. Edge drawing algorithms adopt four basic steps to compute edges of an image. The steps includes: Image smoothing, determination of edge areas and edge directions, computation of edge anchor points and linking of edge anchor points by edge drawing. The goal of image smoothing is to reduce the effect of noise in the image by blurring each pixel with its neighboring pixels and is achieved by a standard 5x5

Gaussian kernel with $\sigma = 1$. Determination of edge areas and edge directions step is to determine potential edge areas, i.e., regions of interest, where the actual edges reside. These areas are found by applying a derivative operator or a high-pass filter to each pixel of the smoothed image. It is possible that this method may be achieved using different operators, howerver in edge drawing a Sobel operator is applied to each pixel in the smoothed image and two gradient values, Gx and Gy are obtained for both vertical and horizontal directions. Gx and Gy are then used to determine the edge areas and the edge directions. If the summation of Gx and Gy is larger than a given threshold, then the pixel is marked as an "edge area" pixel, otherwise the pixel is marked as a "non-edge area" pixel. Moreover, the direction of the edge is determined by comparison between the values of the vartical gradient value Gy and the horizontal gradient value Gx. If Gx is greater than or equal to Gy, then a "vertical edge" is assumed to pass through this pixel, otherwise, a "horizontal edge" is assumed to pass through [18].

When the edge areas and directions are computed, a set of edge point pixels inside the edge areas are computed. These points serve as "anchor points" for the final edge drawing process. To achieve this goal, a raster scan of the edge area image in row-major order is carried out and points of maxima are marked as anchor points. One of the advantages of this step is that there is an option of selecting the amount of detail in the final edgemap [18]. The more anchor points used to start the linking process, the more details appear in the final edgemap. Foristance, to obtain the major (long) edges in the image, a big detail ratio needs to be specified i.e., a ratio larger than 10. For more details and revealing features of the edges, a smaller detail ration is choosen in the edge map such as; 1, 2, 3, 4, etc. The "detail ratio" parameter is used as follows: For a "detail ratio" value of say "k" every Kth row or column is scanned and marked anchor points only if they fall in these rows or columns. Thus, two consecutive anchor points along the same edge will be at least "k" pixels apart from each other. This hole in the actual edge will then be filled during the "edge drawing" process. The final stage in the algorithm is edge drawing by anchor point linking. In this step, the goal is to draw (compute) actual edges by starting at an anchor point and tracing a path to the next anchor point along the same edge area. To guide the path, the gradient map and the edge directions computed in step 2 are deployed [18]. Assume the algorithm starts at a point identified as an anchor point in the middle of an edge. If the edge direction is a vertical edge, the algorithm first link pixels to the north by going over the pixels having the maximum gradient values. This path should end up in the next anchor point to the north. It then start

linking pixels to the south by going over the pixels having the maximum gradient values. This path should end up in the next anchor point to the south. For each marked pixel, the algorithm moves up, and simply look at the 3 neighboring pixels to the north and go to the one having the maximum value. Specifically, if it approaches a pixel with corodinate (i, j) moving up and the edge passing through (i, j) is a vertical edge, then it search for pixels with corordinates $(i, j-1)$, $(i-1, j)$, $(i-1, j+1)$ and simply go to the one having the maximum value. This linking process makes it possible to go over the entire real edge until the next anchor point is encountered [18]. Similarly, as it goes down from an anchor point, it looks at the 3 neighboring pixels to the south, i.e., $(i+1, j-1)$, $(i+1, j)$ and $(i+1, j+1)$, and then goes to the one having the maximum value. This leads to the next anchor point to the south. The actual path is drawn during the linking process and has a good contiguous 1-pixel wide edge. If the edge was horizontal, then the algorithm would have traced a path to the right (east) and to the left (west) of the anchor point. Specifically, going left from (i, j), it would look at pixels $(i-1, j-1)$, $(i, j-1)$ and $(i+1, j-1)$ and goes to the one having the maximum value. Similarly, going right from (i, j), it would look at pixels $(i-1, j+1)$, $(i, j+1)$, and $(i+1, j+1)$ and goes to the one having the maximum value.

While gradient based detectors, such as Prewit and Sobel operators have been researched for parallelism, locating complex edges are inaccurate and their sofware implementaion does not meet real-time requirements [17]. The edge drawing algorithms proposed in [20] have different approach from the classical edge detection algorithms with robustness for real-time processing. The algorithm begins by smoothening the image and then computes some sets of anchor points in the spacial image domain and draws edges between them. The resulting edgemap consists of contiguios one-pixel wide edges with real connectivity [20].

Extracting these important features serves as the basis for object detection and subsequently recognition. The word “recognition” itself has been used in literature to mean interpretation, classification, cognition and inheritent task [10]. Traditionally, the recognition methods heavily depend on global features of the object such as; the geometric and moments features. These traditional approaches have serious drawbacks since they may not be able to detect object in an event of image rotation and background clutter [12]. For these reasons, over the past decades, local invariance features algorithms such as Scale Invariant Feature Transform (SIFT) have been effectively implemented [12]. SIFT has shown great ability to recognise objects, even in the presence of image scaling, rotation, unequal illumination, changes of

viewpoint and to some extents out-of-plane rotation. SIFT-based object recognition algorithms start by identifying keypoints over all scales and image locations, followed by generating a descriptor for keypoints based on the local image patch in its neighborhood. Finally it searches through the database to find a possible match for each descriptor for the final stage of the recognition [12, 15]. The most critical part in the recognition process is the searching for a right match for the object in the database. Sets of trained classifiers which may consist of many stages (e.g., cascaded detectors) compare the extracted object with its different models created in the database.

Each stage of the classifier marks the region defined by the current location of the sliding window as either true or false. “True” indicates an object was found and “False” indicates no object. If the label is “False”, the classification of this region is complete, and the detector slides the window to the next location . If the label is “True”, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as “True”. The stages are designed to reject negative samples as fast as possible. The assumption is that the vast majority of windows do not contain the object of interest. A *true positive* occurs when a positive sample is correctly classified. A *false positive* occurs when a negative sample is mistakenly classified as positive. A *false negative* occurs when a positive sample is wrongly classified as negative. To work well, each stage in the cascade must have a low false negative rate [21]. If a stage incorrectly labels an object as negative, the classification stops, and there is no way to correct the mistake. However, each stage may have a high false positive rate. Even if it incorrectly labels a non-object as positive, the mistake can be corrected by subsequent stages [15,21]. However for a much complicated classification, Artificial Neural Network (ANN) classifier was successfully implemented in [22]. The classifier makes use of Multi-Layer Perception (MLP) algorithm with forward architecture within the input and the output. Though ANN classifier has been shown to have an excellent learning ability and minimum false alarm it has the disadvantage of requiring substantially large amount of training datasets of the object models [22, 23].

One of the critical stages in classifying algorithms is the classifier training which requires a set of positive samples and a set of negative images. A sufficient set of positive images with regions of interest specified to be used as positive samples must first be supplied. Positive sample can be specified in two ways [21]. One way, is to specify rectangular regions in a larger image. The regions contain the objects of interest. The other approach is to crop out the

object of interest from the image and save it as a separate image. Then, it will be also possible to specify the region to be the entire image. Additional positive samples may also be generated from the existing ones by adding rotation or noise, or by varying brightness or contrast. Similarly, a set of negative images from which the function generates negative samples automatically are equally supplied. Negative samples are not specified explicitly [21]. Instead, the classifier funtion generates negative samples from user-supplied negative images that do not contain objects of interest. Before training each new stage, the function runs the detector consisting of the stages already trained on the negative images. If any objects are detected from these, they must be false positives [9, 21]. These false positives are used as negative samples. In this way, each new cascaded stage is trained to correct mistakes made by previous stages.

With sufficient supply of these samples, the number of stages, feature type, and other function parameters to achieve acceptable detector accuracy, may be pre-determined for the training. The choice of the feature detectors to be used for the training may depend on the type of object to be detected. Historically, Haar and Local Binary Pattern (LBP) features have been used for detecting faces [12]. They work well for representing fine-scale textures. The Histogram of Oriented Gradient (HOG) features have been used for detecting objects such as people and cars. They are useful for capturing the overall shape of an object.

For a cascaded classifier the criteria for choosing the number of stages depends on what sort of recognition one is interested at. Since there exists a trade-off between fewer stages and a lower false positive rate per stage or more stages with higher false positive rate per stage, stages with lower false positive rate are more complex because they contain a greater number of weak learners [21]. Stages with higher false positive rate contain fewer weak learners. Generally, it is better to have a greater number of simple stages because at each stage the overall false positive rate decreases exponentially. For example, if the false positive rate at each stage is 50%, then the overall false positive rate of a cascaded classifier with two stages is 25%. With three stages, it becomes 12.5%, and so on. However, the greater the number of stages, the greater the amount of training data the classifier requires. Also, increasing the number of stages, increases the false negative rate. This results in a greater chance of rejecting a positive sample by mistake.

The results of image processing analysis which are generated as digital control outputs, provide the basis for actuating production line mechanisms. The same information are used by industrial robots arm for tracking and stacking of a finished product (Palletization) on pallets

for conveyance to the next level of processes or warehouse for storage. The available palletizers in the market are mechanical and robotic pallatizers. In both types, the key performance criterion is the palletizing throughput, operational flexibility, purchasing cost and level of training required to operate and maintain the palletizer [7, 8]. Mechanical palletizers offer high speed palletizing and low initial costs. However, because these machines are designed for a specific range of products and a limited number of patterns, they offer low flexibility. Manual system reconfiguration is required during product changeover, resulting in long product changeover downtime and significant productivity losses, especially, when production batch sizes become smaller. Frequent set-up errors add to the total downtime and in extreme situations result in costly product and machine damage [7].

Typical robotic palletizers offer high flexibility in palletizing and shorter product changeover downtime. However, they can only achieve low to medium palletizing speed and have a higher initial cost than the mechanical types. During product change over, there are still some mechanisms that need manual adjustments, which frequently results in set-up errors that add to the total downtime, and can cause significant losses due to potential product and machine damage in industries [8]. In our design a simple mechanical device to provide orientation correction is proposed. It can be used in connection with the robot palletizer for higher performance and flexibilty. The device is based on the orientation information of the product obtained from the image analysis. It is made of two equal length parellel links with degree of rotaion horizontally on the conveyor's plane. In both ends, the links are joined by bars in the conveyors direction and two similar asynchronous motors provide the necessary torque to derive the device. The trajectory of the mechanism is in such a way that it travels in the direction of the on-coming product and it is able to stretch to meet the product for re-orientation. The tacile pressure sensor on the other wall of the conveyor provides the feedback on the extent to which the mechaninism can go. With the fully oriented product arriving the collation point, robot can easily be programmed from the home position to pick and stack the product in a place of interest without having to deploy its own vision system which is costly and time consuming. The mechanism can provide enough torque to orient heavy product stacked palletes like cements.

CHAPTER TWO

2. PROBLEM STATEMENT

2.1. Formulation of the Problem

Machine vision is rapidly becoming an essential part of modern day's industrial applications with attendants' benefits of improved performance and ability to handle complex industrial processes. The vision system grapples with some challenges. Some of these challenges include huge computations and processing of frames of images captured by high-resolution image sensors [1, 3]. A great deal of speed is required to accomplish a task and the need becomes much higher when the process is performed in real time. GPP-based processing has been the norm for many years; however with the increasing demand for higher processing speed for color images, GPP processing has serious drawbacks, especially, when implemented on an embedded system [3, 6]. QCP are among the industrial applications that extensively deploy the services of vision system to extract features, positions and orientation information about online production. The process is similarly affected by delay in outputting the results from the image processing hardware to the robot controller. This increases the total downtime of the production with low efficiency consequences [7, 8].

QC application is an important part of the industrial process of any manufacturing industry. It is essentially a process in which production is reviewed, inspected and controled to ensure the quality of all factors involved in production. This approach places an emphasis on some key aspects of production that are worth noting before products are being released. The controls include product inspection, where every product is examined visually by human or using machine vision to uncover defects such as; cracks or surface blemishes in production and allow for the decision to be taken to deny product release. Due to the need for speed and accuracy in sorting the defect products, machine vision has been the preferred choice [7, 21]. The vision system, basically, depends on the image processing algorithms to differentiate good production from the defective ones. In turn the image processing algorithms, specifically, extract major features from the image of the products being captured by the camera and compare it with the pre-stored image of the same product considered to be good product. A major deviation from the features of the product stored in the database may

indicate defect. In most cases the image processing algorithms are run on GPP which may not meet real-time processing requirements and, sometimes, lead to intermittently stopping the inspection for a while and wait for a result from the vision system and continue the next process afterwards. Some of the major causes of such delays are the high computation density of the image processing algorithms and unpredictable scheduling of the operating system [4, 5, 21]. These delays increase the total production time and offcoure the production cost.

On the other hand, palletizers are designed to automatically load cartons onto palletes and wrap plastic film around them for shipping or storing in warehouses in manufacturing industries. Pallets are normally wooden-made flat type containers which are used for loading and packaging and can also be used for bulk storage of stock in an open plan floor area. Productivity of a palletizer system is assessed by system throughput which is defined by the hourly rate of transactions that the palletizer can perform [7]. Two major types of palletizers are commonly found in the industries today. They include the mechanical palletizers, and robotic palletizers. Both types have their advantages and drawbacks. Basically, the mechanical palletizer is based on pre-determined motion and controlled by sensory inputs and hence does not use the vison system. Robot palletizer achieves its result by teaching the robots a set of coordinate points of the work-piece and its required trajectory. The more complex and advanced form of this robotic palletizer uses vision system. In both, the key performance criteria are based on the palletizing throughput, operational flexibility, purchasing cost, and level of training required to operate and maintain the palletizer [8]. Mechanical palletizers are relatively cheap with high operational speed. Their major setback is the lack of flexibility of operations (i.e., they are designed for localized operations). Moreover, manual system reconfiguration is required during product changeover, resulting in long product changeover downtime and significant productivity losses, especially when production batch sizes become smaller. Frequent set-up errors add to the total downtime and in extreme situations can cause costly product and machine damage. Typical robotic palletizers offer high flexibility in palletizing and shorter product changeover downtime [7].

However, they can only achieve low to medium speed palletizing and have a higher initial cost than the mechanical types. During product changeover, there are still some mechanisms that need manual adjustments if the robot pallatizer is based on trajectory teaching, which frequently results in setup errors that add to the total downtime, and can cause significant losses due to potential product and machine damage [7]. For the more advanced robotic palletizer that uses a vision system, the major challenge is the image processing algorithm

execution time. The algorithms are tasked with recognizing the products and their corodinate points. Since these algorithms are implemented same way as in Quality Control same problems are experienced.

Attempts were made to curtail some of the challenges emanating from the software-based implementation of the image processing algorithm on the GPP processor. Most of these scholarly works proposed the implementation of these algorithms on a standalone FPGA processor. FPGA processor, apart from being a dedicated hardware, is immuned from the unpredictability of the operating system scheduling and has ability for true parallel processing and data pipelining. Z. Guo *et al.* [4] demostrated a parallel processing construction of Sobel edge detection enhancement algorithm on FPGA hardware, which can quickly get the result of one pixel in only one clock period. The algorithm is designed with a FPGA chip called XC3S200- 5ft256, and it can process $1024\times1024\times8$ gray scale image successfully. The design can locate the edge of the gray image quickly and efficiently. It uses 3×3 convolution kernels of gradient approximation matrix of the sobel operator to process $1024\times1024\times8$ gray scale image. The architecture of the FPGA was divided into four modules for implementation. The modules include: 3×3 pixel generation module, Sobel enhancement operator module edges control module and binary segmentation [4].

In a similar way, I.Yasri, et al. [3] designed and implemented gradient- based edge detection algorithm using Sobel operator on FPGA. The Sobel Edge Detection operator is controlled by Finite State Machine (FSM) which executes a matrix area gradient operation to determine the level of variance through different number of pixels and display the result on a monitor. The whole process was performed in the hardware level and implemented on an altera field programmable gate array platform. The result shows good performance of edge detection with 27MHz clock operation which is able to detect within just 2ms [3].

A. Sultana and M. Meenakshi designed and implemented a real-time validation of image binarization process using weight based clustering algorithm [2]. It uses the clustering property of neural networks on FPGA hardware. The developed algorithm was divided into two functional blocks. The first block is used to obtain the threshold value for the image frame and the second block to apply the threshold value to the frame. The parallelizm and the simple hardware component of both blocks make the approach suitable for real-time applications. They further demostrated that the performance remains comparable to the Otsu technique which is frequently used in off-line threshold determination. Results from the

proposed algorithm are presented for numerous examples, both from simulations and experimentally using the FPGA [2].

A. Bochem, et al. [26] presented the implementation and evaluation of a computer vision task on FPGA which is an experimental approach for an application-specific image processing problem. It provides results about gained performance and precision compared with similar solutions on GPP architectures. The design adressed problems associated to BLOBs in a continuous video stream and computation of their centroid most of which existing solutions are realized on GPP platforms, where resolution of the image material and sequential processing define the performance. The evaluation compares precision and performance gain against similar approaches on GPP platforms. They demostrated different concepts for BLOB detection and showed the implementation of one common method for BLOB detection, including design problems and performance evaluation [26].

In this work, a real time image processing design is proposed to be implemented on a fast dedicated FPGA processor to address some of the drawbacks associated with the GPP image processing implementation. The methodologies to be adopted would initially start with the software implementation of the image algorithm that will meet the QCP requirements in MATLAB programming environment. The image processing algorithms would essentially encapsulate CCA of the 4-neighborhood pixels of an image data to extract BLOBs and some vital features. Viola-Jones cascaded classifier will be used to sufficiently recognize the objects based on the extracted geometrical features. The software version of the algorithm would be converted to Simulink Model-based design from where HDL codes can be automatically synthesized for deployment onto the FPGA hardware. The hardware architecture of the FPGA is designed in such a way that it conforms to the model-based Simulink version. On the other hand a different approach is proposed in handling the physical actuation of the palletizer. Here, a mechanical palletizer is designed. The palletizer has information sourcing from the same image processing algorithm and can work with the robot palletizer for improved production speed. It is modeled as a four-bar mechanism with a properly planned trajectory to reorient and align products. It is based on the image processing outputs. Since both processes are aimed to work harmoniously on the conveyor belt, timing and synchronization of the various units become paramount. Computer simulation and performance analysis of the conveyor and four bar mechanism is deduced. Links synthesis and trajectory planning of the four bar mechanism also conducted.

2.2. Objectives of the Thesis

The objectives of this thesis are itemized as follows:

1. Designing a system that can take care of QCP problems
2. Applying image processing technique and integrate it with the proposed system
3. Performing numerical simulations for the proposed system
4. Applying the proposed system in a real-time application to examine its performance
5. Conducting performance analysis and troubleshooting
6. Conclusions and recommendations for future work

CHAPTER THREE

3. CONCEPTUAL DESIGN AND MATHEMATICAL MODELLING

3.1. Image Processing Algorithms

Fundamentally the algorithms concern with the preprocessing of image frames and extracting some vital features from them via applying some operations on the image elements (pixels). The two major operators being deployed here include Morphological operation and CCA operations on the binary version of the original image. Morphological operations play a vital role of preprocessing through estimating the background and foreground of the image which provide means for removing uneven brightness over the entire image by subtracting the background from the rest of the image. A more complex combination of Morphological opening followed by Morphologial closing operations could lead to revealing more useful features of the image such as edges. On the other hand, from the binary image, CCA computes the geometrical features of the binary objects within the image frame. The computed geometrical features such as perimeter, area, centroid, and more, may combine well in the cascaded classifier algorithm to train the algorithms for more accurate object detection and subsequent recognition.

3.1.1. Morphological Operation

Morphological operation, in the context of computer vision, refers to the operation carried out in two dimensional (2D) spatial domain of an image to describe the properties, shapes and area of objects on the image plane [11]. The operation becomes useful in the processing task of a set of points to reveal vital features of an object. The set of points in the image spatial domain represents an object. Four major Morphological operators include; Dilation, Erosion, Opening and Closing. While Dilation increases the size of the image, Erosion thins the image [21]. Morphological Opening and Closing are compound operators which allow to fill an inner hole or help in getting rid of a small fragment on the image, respectively [21, 24]. The two major inputs to the Morpological operators are the input image and, a specially created image called, Structural Element (SE). The choice of the SE is arbitrarily and mostly depends

on the type and the nature of the operation intended. In most instances, SE is much smaller than the processed image since it is a form of area description of sets of binary points (object) within the image plane [21].

During the operations all the pixels in the output image are initialized as zero pixels. SE combines with every pixel in the input image by logically passing its center through that pixel. In Dilation, when the center of SE passes through an image pixel, the entire SE logically combines with that pixel and alters its value. The corresponding new value of the pixel is written into the same position in the output image. For Dilation the operation is a logical addition which increases the size of the objects in the output image while in Erosion it is a logical disjunction of the central SE pixel and the corresponding pixel in the output image. In binary image, all objects smaller than SE will be deleted, hence it is a means for removing noise and image background estimation [21]. The problems associated with Dilation (thickening) and Erosion (thinning) of the original image can be corrected using the compound operators Opening and Closing. Opening is Erosion followed by Dilation to restore the original image size after being processed. Morphological Closing is the direct opposite of Opening. In both cases, the operation can only be applied once as performing more has no effect on the output image [21, 25]. Subtracting the eroded image from the original input image reveals the edges of the image. Equations (3.1)-(3.4) represent Dilation, Erosion, Opening and Closing, respectively, on an input image $f(x, y)$ that produces output image $g(x, y)$.

$$g(x, y) = f(x, y) \oplus SE, \tag{3.1}$$

$$g(x, y) = f(x, y) \Theta SE, \tag{3.2}$$

$$g(x, y) = f(x, y) \circ SE = (f(x, y) \Theta SE) \oplus SE, \tag{3.3}$$

$$g(x, y) = f(x, y) \bullet SE = (f(x, y) \oplus SE) \Theta SE. \tag{3.4}$$

Where Θ and $\oplus$ are the Erosion and Dilation operators, respectively.

3.1.2. Connected Components Analysis

CCA algorithms are applied to colored, grayscale and binary images [26]. Initially, the entire image is scanned, pixel by pixel, rowwise and columnwise from the top to left and down to the right bottom corners. If any group of connected pixels shares equal or similar designated pixel intensity values are labeled as object in the output image. Each set of distinct connected pixels have a distinct label in the output image. In a binary image, any pixel encountered during the scan and found to have value of '1' is an object pixel while '0' value indicates non-object pixel. In grayscale a certain range of intensity value (i.e., 0-50, 60-80) can be assigned to indicate object pixel and any pixel with intensity within that range is written as an output object. Modification to the algorithm makes it possible to operate on RGB colorspaced images at different level of connectivity i.e 4 and 8 neighborhoods connectivity.

In a binary image with 4 neighborhood connectivity, as adopted in this thesis, the connected component labelling operator scans the image by moving along its colums untill it finds the first pixel at point say K, whose logical value is '1'. The operator places its center pixel at that point. The pixel at point K is labelled and written in the output image and get burnt in the input image (i.e., coverted to zero), indicating that it has been scanned. The four neighbors of the burnt pixel at point K are examined (starting from the pixel right of K clockwise) for three possibilities. If all the four neighbors have logical '0' values, the pixel in question is a disconnected object and has a unique output label. If any of the neighbors is a logical '1' it will also be burnt and assigned the same label as the pixel at K, whereas, if all the neighboring pixels are logical '1' they all get burnt and get the same labelling as pixel K. The algorithm continues untill all the pixels are processed and second scan is performed to sort out connected objects into equivalent classes each with distinct label. The algorithm is often referred as Recursive Grass-Fire algorithm [21, 26]. It is obvious that some of the extracted connected objects may have very small area of size of few pixels. These objects are undesirable and Morpological Opening may precede the operation to ensure that only BLOB makes it to the algorithm.

Similarly, the 8-neighborhood connectivity operates in a similar way to 4-neighborhood connectivity. They only differ in the sense that 8-neighbors of a white pixel are examined in

contrast to 4 neighbors. Though the 8-neighborhood connectivity is more robust and accurate, it comes at a cost of high computational density [21]. The choice of 4-neighborhood connectivity is due to the fact that it is faster and suffient for the applications in review. Figures 3.1 and 3.2 show 4- and 8-neighborhood connectivity operators computing BLOBs from binary images.

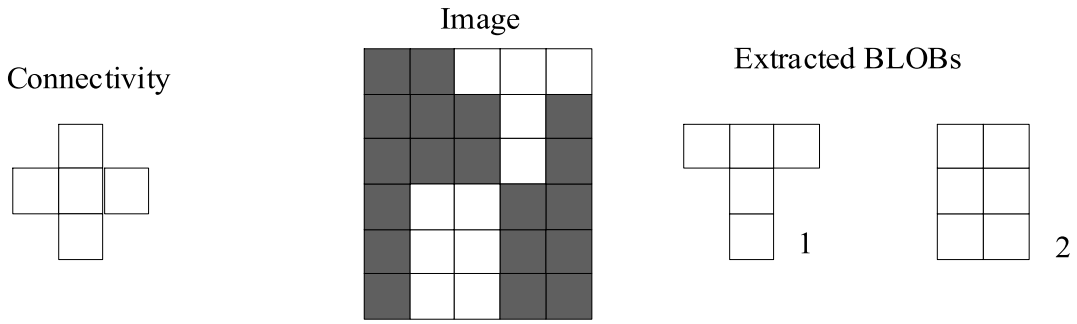


Figure 3.1: 4-neighborhood connectivity operator.

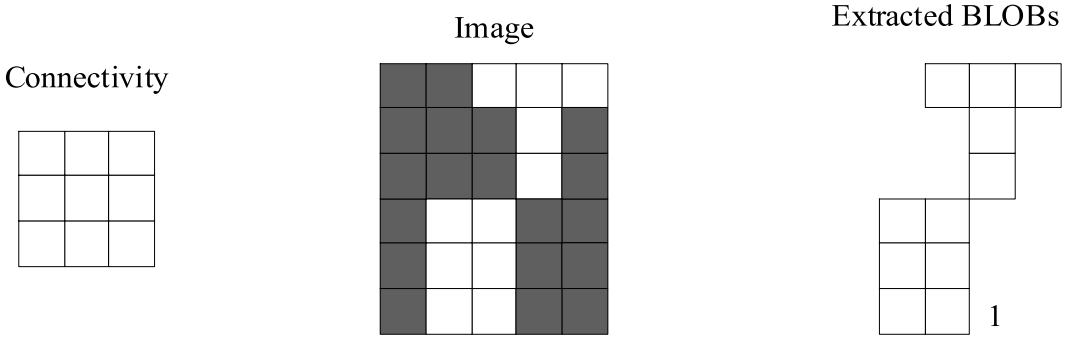


Figure 3.2: 8-neighborhood connectivity operator.

3.1.2.1. Geometrical Features Extraction

Each labeled connected component object is treated as a unique BLOB and stored in a vector with its labeled values and corresponding indices. Considering a BLOB whose minimum and maximum x and y coordinates of its pixels are denoted by $x_{\max}$, $y_{\min}$, $x_{\min}$ and $y_{\max}$ respectively. If the total number of pixels within that BLOB is N which is also equivalent to the BLOB's area, (3.5) and (3.6) compute both the Bounding Box area (BB) and Centroid (C) of the BLOB.

$$BB = (x_{\max} - x_{\min}) \times (y_{\max} - y_{\min}) \tag{3.5}$$

$$C = \left[\frac{1}{N} \sum_{i=1}^{N} x_i, \ \frac{1}{N} \sum_{j=1}^{N} y_j \right] \tag{3.6}$$

where *i* and *j* denote the ith row and *jth* column, respectively.

The orientation of the BLOB is the angle in degrees ranging from negative 90^0 to positive 90^0 degrees ($-90^0 - 90^0$) along the x-axis and the major axis of an ellipse that has the same second moment as the BLOB being investigated. Fig. 3.3 shows a connected binary object with an ellipse of the same second moment. The orientation is the angle between the horizontal line and the ellipse's major axis.

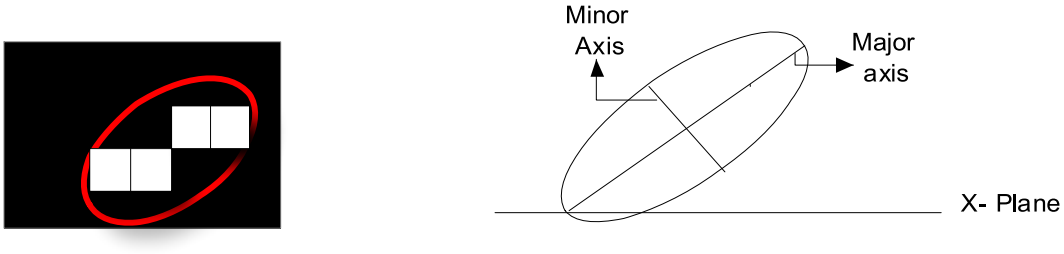


Figure 3.3: BLOBs with an ellipse of the same second moment.

The extracted goemetrical features of the BLOB are housed in object feature vectors which not only provide statistics about the BLOB but also become a simple classifer to identify the type and nature of the BLOB of interest. For example, circularity measure of BLOB in (3.7) can help to distingush between circular and non-circular objects.

$$Circularity = \frac{\sum counter\ Pixels\ (Perimeter)}{2\sqrt{\pi \times BLOB\ Area}}. \tag{3.7}$$

3.1.3. Object Recognition

For an object whose aspect ratio does not vary significantly, rule-based classifiers are efficient. However if the aspect ratio varies significantly artificial intelligence classifier such

as ANN classifier will suffice [22, 27]. The most familiar amongst the rule-based classifier is the cascaded one using Viola-Jones algorithm [27]. The powerful tool of the algorithm is its ability to identify keypoints on the object and the unique features surrounding these points. Feature algorithms such as; HAAR and LBP are historically used for detecting face while Histogram of Oriented Gradient (HOG) algorithms are suited for capturing the overall shape of an object such as cars, boxes and people. The cascaded classifier is made of stages and each stage consists of an ensemble of weak learners. The weak learners use the feature vectors to make decision and the weighted average decision of the learners represent the classification status of that stage [21]. Each stage slides its detection window over the entire image and declares the image positive if the target object is found or negative for a non-target image. If the classification of the first stage turns out positive the image is passed to the next stage of the classifier which slides over its window for further testing. If at any stage, an object is classified negative, the classification stops there and does not proceed to the next stage. An object is reported positive only at the current window location in the final stage of the classifier when the detector classfied the image region to be positive [21].

Three scenerios of the classification status exist; True Positive, False Positive and False Negative. True Positive occurs when the object is correctly classified; False Positive is when non-target object is mistaken for positive (target), while False Negative happens when target object is classified as non-target. To optimise the classification and reduce high False Negative rate, high False Positive rate can be set at each stage to allow passage of suspected target to the subsequent stages. This is understandable since if a particular stage incorrectly labels a target object as negative, the classification stops and there is no other way to undo or correct the mistakes. To train the classifier sufficient set of positive and negative samples of the target object must be supplied to the classifier. The negative samples, typically, consist of background images associated with the target object and the function automatically generates more negative images from the supplied ones. An error message would be generated if at any stage of the training, the samples used are insufficient. A high True Positive rate at each stage may ensure that each stage has adequate positive samples for training. The number of positive samples (PS) required to train each stage is computed below [21].

$$PS = floor\left[\frac{Total\ Positive\ Samples}{(1+(Num\ Cascaded\ Stages-1)\times(1-True\ Positive\ Rate))}\right]. \tag{3.8}$$

3.2. FPGA Architectural Design and Implementation

FPGA technology is rapidly becoming a powerful alternative for software algorithms' implementation to the traditional GPP [5]. FPGA processor is made up of a silicon chip with reprogramable digital circuitry. The internal circuitry is constituted into thousands of logic block units known as Configurable Logic Blocks (CLB). Between the CLBs there are re-configurable wiring circuitries that can be used to logically interconnect all the CLBs. While GPP processor takes time executing each function by, sequentially, executing set of instructions, in contrast FPGA has true parallel processing where a given set of CLB configurations can execute functions at the same time independently without sharing any system resources for any given task [2]. The application of FPGA in real-time image and video processing algorithms has compelling benefits. Apart from true parallel execution and high computational density, it is possible to use line buffers to concurrently execute sets of streaming data (pipelining) to further speed up the execution time.

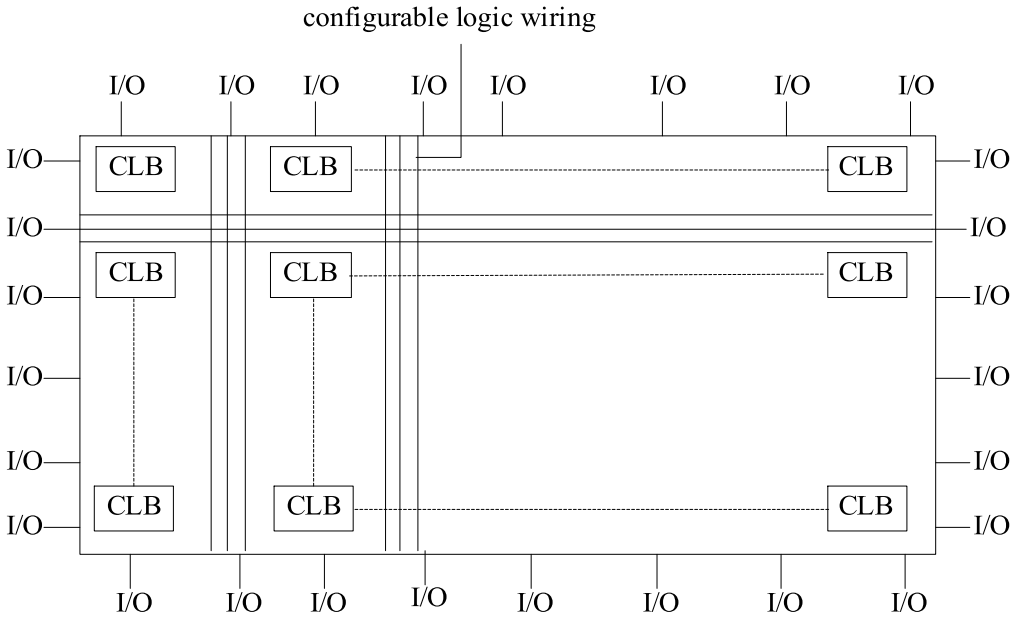


Figure 3.4: FPGA Chip.

3.2.1 FPGA Design Architecture

The Hardware architecture of the FPGA is as shown in Fig. 3.5. It has been grouped into five system blocks. It is, possible, nowadays to do the preprocessing like gray extraction and denoising on the cameras through fine tuning the camera setting parameters. The critical

blocks are the serial data extraction, thresholder, and BLOB and classifier blocks. In the serial data generator four groups of shift registers are used. Each block of shift registers is internally made up of four '1' bit register. Between two blocks of shift registers, a First-in First-out (FIFO) line buffer is used. Each FIFO is capable of caching 3 bits of image data shifted out of the preceeding shift register block. A total of 3 FIFO and 4 shift register blocks would be required to creat 4x4 image windows in 4 clock cycles. The FIFO is generated by a dual-port RAM instead of FIFO IP [4]. In each 4×4 window of image data created, a new center pixel is computed. At each clock signal, the data advances one step to the right by the shift registers. The 4×4 data window is processed and streamed to the blocks of its right. All 4×4 windows have a modified center pixel value. This procedure continues until all the image pixels are processed. At the output, the original image is used as a reference together with the modified center pixel value of the streamed windows to reconstruct the new image containing the BLOBs. Control signal is required to stream the 4×1 column of data into the algorithm. The 4×4 window allows the 4 neighborhood connectivity algorithm to check the adjacent pixels of the pixel currently operated. Since 4 clock cycles are used to create 4×4 image and compute the first center pixel, additional control signal at the output of the image is required to indicate the validity of the center pixel. Fig. 3.5 shows the architecture while in Fig. 3.6 the bottom shift register block holds the oldest image data.

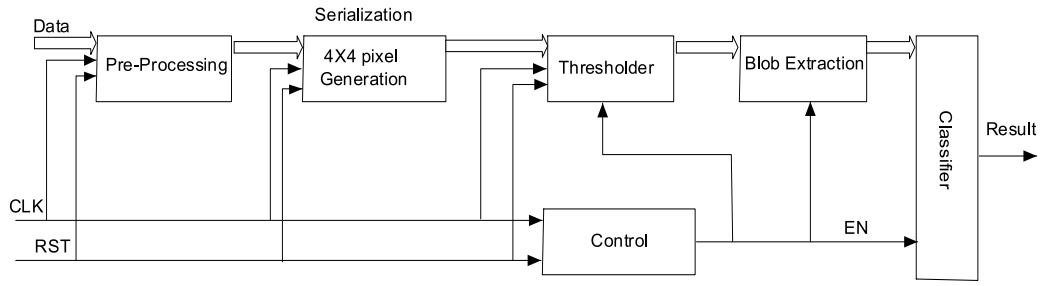


Figure 3.5: FPGA Architecture.

3.2.2 FPGA Implementation of the Algorithm

Hardware implementation of algorithms in FPGA is traditionally written and compiled HDL such as; Verilog and VHDL programming languages. FPGA hardware integrated development enviroment like QUARTUS ii provides the enviroment for coding and compilation of the HDL programmes into the FPGA hardware for field implementation.

However, to accomplish the image processing algorithms on the hardware, a special design and reconfiguration of the hardware architecture need to be done. This is due to the fact that the HDL codes generation may be too long or complex and sometimes incapacitated with inadequate library functions for image processing [21]. The present day architrecture of the FPGA works on a streaming data in contrast to the 2D image formats. With a limited working memory of FPGA 2D processing is not supported [3]. To avoid these limitations and take advantage of pipelining and true parallel processing nature of the FPGA, a series of step is adopted.

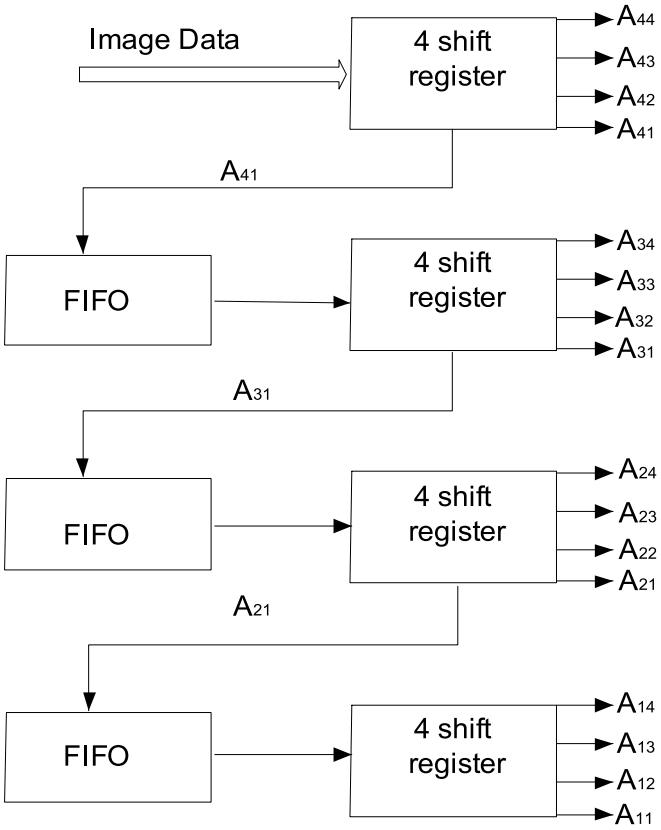


Figure 3.6: 4×4 pixel Generation Architecture.

The design workflow of the algorithm, leading to its implementation on the FPGA begins with the software development of the algorithm in the MATLAB programming enviroment. Based on the functionality requirements, the system model is built in SIMULINK from the MATLAB codes. Fundamentally, the MATLAB algorithms run on the GPP while the SIMULINK version is targetted for FPGA. Due to this disparity the model has to be re-

analyzed and further optimized to meet up with the real-time requirements. The next stage elaborates the design and this becomes paramount to avoid obvious constraints such as processing of the 2D image data, avoidance of complex or unsupported control flow functions like 'WHILE' and 'BREAK' statements [3]. Once the model is elaborated and conformed with what the original algorithms are intented for, HDL codes are automatically generated from the elaborated simulink model. The HDL codes create a Tranfer Register Language (RTL) with directly compiling, mapping and routing the RTL codes into the FPGA hardware. From the RTL codes, HDL 'Testbech' would be created to cosimulate the behavior and performance of the HDL codes on the hardware with reference to the original SIMULINK model using FPGA implementation tool software such as MODELSIM. The last stage, is the ability to affect on-the-field (field programmable) changes, adjustement, verifications and improvements on speed and memory utilization. Fig. 3.7 depicts the implementation design workflow.

3.2.3 Data Serialization

The image data needs to be reconstructed to make it suitable for HDL codes generation targetted for implementation on FPGA. This is because the data type in the algorithm may contain double precision data, strings and some control flow constructs that do not map well on the FPGA processor [16, 21, 31]. A part from these constructs, the hardware does not support 2D processing due to the limited memory [16]. For a 1024x1024 size image of RGB color-spaced the bit depth will be $1024\times1024\times8\times3$ (18KB) which is a large data and will occupy a significant chunk of system memory and line buffers which are in limited supply as per FPGA is concern. Serialization makes it possible for the data to be broken into streaming data before being fed to the chip. The serialization of the image data depends on how many of the image pixels needed to be available for the algorithm computation and how much is the memory and line buffers are available on the chip to stream the data.

In this design, the serialization happens at the Simulink model based design similar to the architecture depicted Fig. 3.5. The critical parts of the algorithm are the thresholder, BLOB computation and detection blocks. Instead of scanning the entire image by the four neighborhood connectivity of the grass fire algorithms adopted here, a modification is made. The image data is converted to serial data columnwise. Out of this, streamed data it is broken into 4×1 column of pixels which is fed to the algorithm.

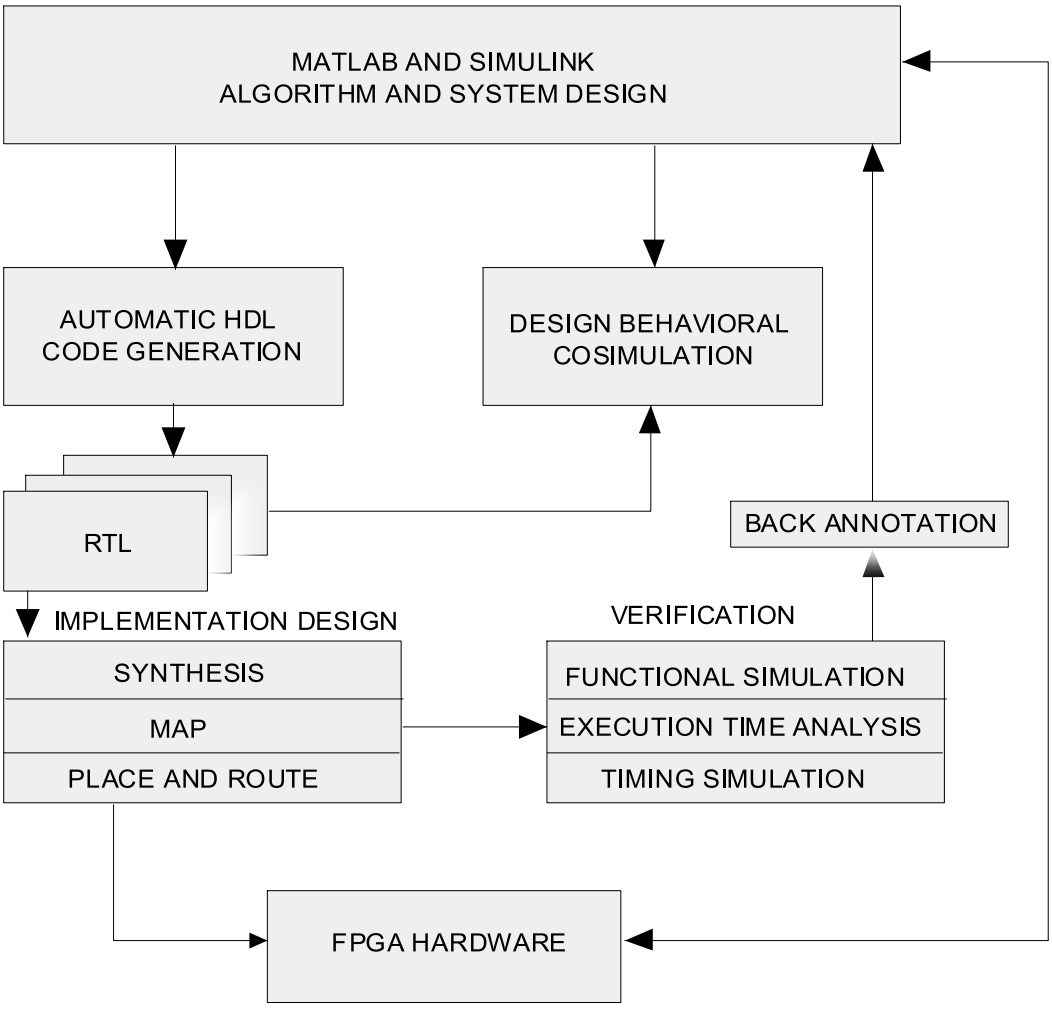


Figure 3.7: Implementation Stages.

3.2.4. CCD Camera and the FPGA Speed Analysis

The choice of the right image sensing device, processing hardware and image processing algorithms play a vital role in defining the overall speed and efficiency of the entire system. In the proposed system, Sick IVC-2D1111 smart camera is intended to be used. Basically, the camera consists of cells of CCD image sensors. Each cell of the CCD image sensor is an analog device. When light strikes the chip, it is held as a small electrical charge in each photo sensor. The electrical charges are converted to voltage, one pixel at a time, as they are read from the chip. Additional circuitry in the camera converts the voltage into digital information

[32-36]. The image sensors have processing speed of 150 MHz with 15 MB flash and 64 MB RAM memories. The camera has frame rate of 30 FPs with 640×480 $pixels$ image resolution.

Moreover, it supports past transmission of image data with a fast Ethernet (10/100 MB) with four control inputs and a trigger. With good feaures, and ideally reasonable, we also designed it to capitalize on the trigger input to intermittently take a snapshot only when the sensor $S0$ senses object. This strategy significantly reduces the number of frames required to be processed by the algorithm there by speeding up the process. On the FPGA harware, where the algorithms reside, the architecture proposed in 3.2.1 is optimally designed to achieve an enhanced speed. For instance, the used four 4-bits shift registers allow processing of 4 image pixels per clock on four different CLBs in parallel.

3.3. Mechanical System Modelling

Primarily the mechanical system consists of two major multi-body subsystems. The belt conveyor including its associated components and the orientation mechanism. In both subsystems electrical motor is a crucial player in deriving the mathematical models, static and dyanamic simulations of the process. It generates the necessary drives (Torque and Speed) that set the entire systems to motion. Therefore, a study of the motor characteristics at different situations becomes the basis for modelling and simulation of the entire process. The motor converts the input electrical energy into mechanical energy at the output. The mechanical energy appears as a twisting force (Torque) at the shaft of the motor and causes it to rotate about an axis [29]. The shaft torque is directly coupled to the conveyor's drive pulley or roller via gearings and chains to drive the conveyor.

3.3.1. Mathematical Model of the Belt Conveyor

Belt conveyor is a combination of mechanical and electrical systems; it can be seen to consist of three major parts. These parts are; the drive motor, mechanical subsystem and control system. The mechanical subsystem consists of belt, shaft, drive pulleys, idlers, etc. The series of processes from the electrical power input up to the torque at the pulley can be related in time domain as time varying quantities [29, 30].

The dynamic characteristics of the motor are described by its inertia, J and calculated by partial differentiation and deduced based on the motor's equivalent circuits and classical laws of motion [29]. Consider an electric motor equivalent circuit diagrams as shown in the figures below.

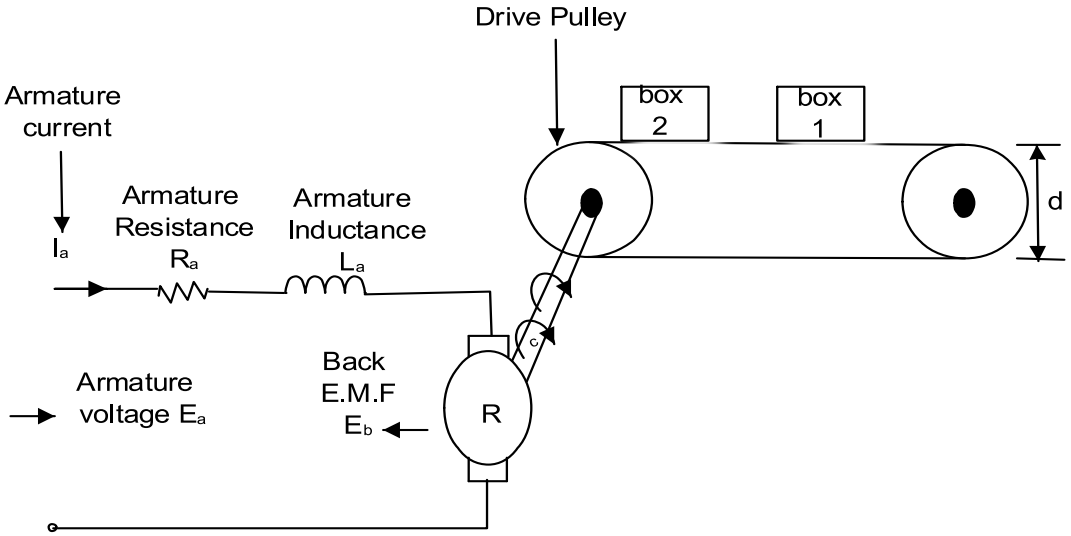


Figure 3.8: Schematic Diagram of the Belt Conveyor with Rotor Circuits.

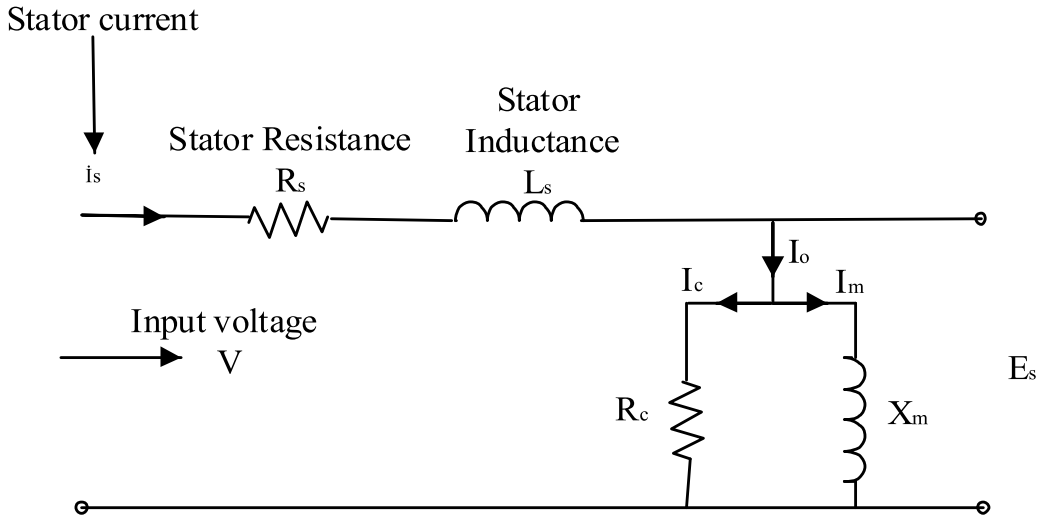


Figure 3.9: Stator Equivalent Circuit Diagram

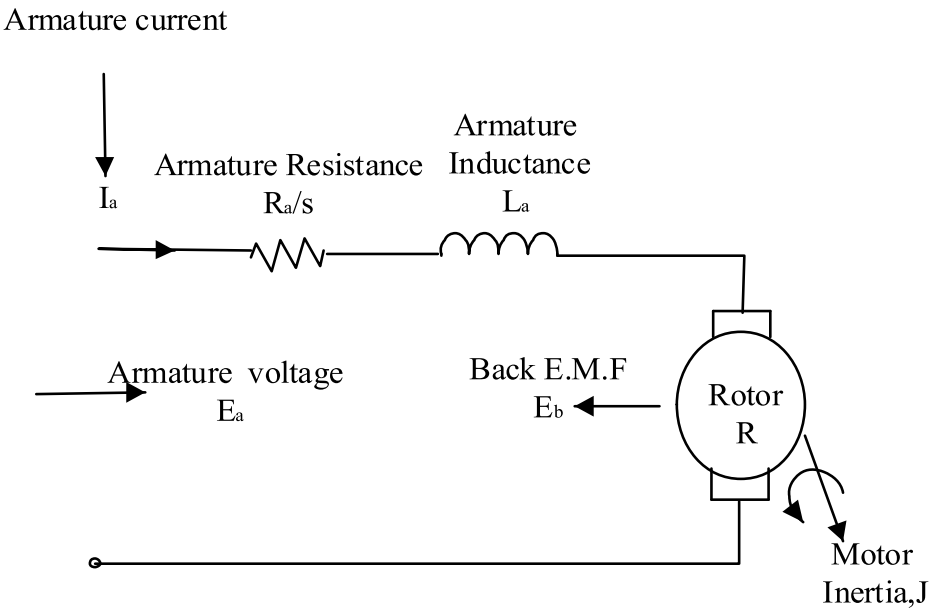


Figure 3.10: Rotor Equivalent Circuit Diagram.

$$E_a(t) = I_a(t)R_a + L_a\frac{dI_a(t)}{dt} + E_b(t), \tag{3.9}$$

$$T_e(t) = T_l(t) + J\frac{d\omega(t)}{dt}, \tag{3.10}$$

$$E_b(t) = K_E \times \omega(t), \tag{3.11}$$

$$T_e(t) = K_T \times I_a(t). \tag{3.12}$$

From (3.9) and (3.10) the following equations can be deduced;

$$\frac{dI_a(t)}{dt} = \frac{E_a(t)}{L_a} - \frac{E_b(t)}{L_a} - \frac{I_a(t)R_a}{L_a}, \tag{3.13}$$

$$\frac{d\omega(t)}{dt} = \frac{T_e(t) - T_l(t)}{J}, \tag{3.14}$$

where $T_e(t)$ and $T_l(t)$ are the electrical and load torque, K_E is the electromotive force (e.m.f) constant, K_T is the torque constant. $E_a(t)$, L_a and R_a are the armature's voltage, inductance and resistance respectively. $E_b(t)$ is the armature back e.m.f. The angular velocity at the

motor's shaft ω is expressed in terms of the number of revolution per second (n) of the motor's shaft.

$$\omega = \frac{2\pi n}{60} = \frac{\pi n}{30}. \tag{3.15}$$

The output torque, T_o, developed at the shaft of the motor depends on the output power, P_o, and the angular velocity of the motor.

$$T_o = \frac{P_o}{\omega}. \tag{3.16}$$

Everytime the roller spins one revolution; the conveyor will move a linear distance equivalent to the circumference of the roller. Hence the linear distance traveled by a point on the conveyor per revolution is equal to the circumference (πd) of the roller. The corresponding linear speed (V) is given by:

$$V = \pi dKn, \tag{3.17}$$

where d is the diameter of the roller and K is the ratio of the shaft diameter to that of the roller.

In short conveyors, the belt flexibility does not significantly affect the behavior of the belt during the starting and normal operations; hence static design model is adequate [30]. However, for a long and bulky conveyor, dynamic models must be used due to some potential problems such as; excessive belt tension, structural load, belt sagging, slippage and increased kinetic friction due to increased weight. The total kinetic friction resulting from the increased belt weight and the load acts to oppose the torque providing the drive. The resulting kinetic friction results in decreasing speed, increasing tension and belt slip. The total kinetic friction force acting on the system with the accelaration due to gravity g is given by:

$$F_k = \mu_k (m + m_o) g \cos\theta. \tag{3.18}$$

In which ($m + m_o$) is the total mass of the belt and the loads, μ_k is the coefficient of kinetic friction and θ is the angle between the conveyor and the horizontal plane. For horizontal plane conveyor $\cos\theta = 1$.

3.3.2. Modelling of the Orientation Mechanism

Apart from the belt conveyor, another important part of the mechanical multi-body subsystem is the orientation device. The device is fundamentally a four-bar linkage mechanism with a well modelled and planned trajectory to achieve its purpose. It is a movable chain with four linkages connected in series by four joints. Each joint has one degree of freedom and could either be *Revolute* (hinged joint) or *Prismatic* (sliding joint). In a planar quadralateral linkage, as adopted in this design, one link is fixed and is designated as the fixed link or *frame* (r_1). The other two links connected to both ends of the frame are the *input* (r_2) and the *output* links (r_4), respectively. The link connecting the input and the output is the *coupler* link (r_3). For a proper understanding of the mechanism operation and the desired trajectory on the conveyor's plane, the links lenghts have to be synthesized. Also, the angular positions speed and accelaration must be deduced. Consider Fig. 3.11 below where P is any spatial point that defines the trajectory of the coupler link whose distance from joint B is given as r_p.

3.3.2.1. Displacement and Velocity Computation

Different types of a four-bar mechanism configuration exist and aim to accompolish a specific task. For a singular linkage configuration, in which the sum of any two links equal to the sum of the remaining links, the following equation holds for the closed loop links [29, 31].

$$r_2 + r_3 = r_1 + r_4. \tag{3.19}$$

Incorporating the respective angular positions of the links in complex form into (3.19) one can get;

$$r_2 e^{(i\theta_2)} + r_3 e^{(i\theta_3)} = r_1 e^{(i\theta_1)} + r_4 e^{(i\theta_4)}, \tag{3.20}$$

Where θ_1 is the angle between horizontal axis and the fixed link, θ_2 is the angle between horizontal axis and the input link, θ_3 is the angle between the horizontal axis and the coupler link and θ_4 is the angle between the horizontal axis and the output link.

Assigning arbitrary length values for all the four links and starting from the home position, frame link angle θ_1 is constant. Taking θ_2 as a reference (independence variable), coupler

angle θ_3 and the output angle θ_4 are the only unknown dependence variables. Rearranging (3.20) becomes;

$$r_3 e^{(i\theta_3)} - r_4 e^{(i\theta_4)} = r_1 e^{(i\theta_1)} - r_2 e^{(i\theta_2)}. \tag{3.21}$$

The right hand side of the equation is made of known parameters and can be expanded into a complex number $X + iY$ as follows:

$$r_3 e^{(i\theta_3)} - r_4 e^{(i\theta_4)} = X + iY. \tag{3.22}$$

Expanding (3.22) and equating real parts and the imaginary parts we have;

$$\cos\theta_3 = \frac{(X + r_4 \cos\theta_4)}{r_3}, \tag{3.23}$$

$$\sin\theta_3 = \frac{(Y + r_4 \sin\theta_4)}{r_3}. \tag{3.24}$$

Taking the square roots of (3.23) and (3.24) and summing them together we have:

$$\theta_3 = \tan^{-1}\left[\frac{Y + r_4 \sin\theta_4}{X + r_4 \cos\theta_4}\right], \tag{3.25}$$

and,

$$\theta_4 = \tan^{-1}\left[\frac{Y}{X}\right] \pm \cos^{-1}\left[\frac{(X^2 + Y^2 + r_4{}^2 - r_3{}^2)}{(-2X.r_4\sqrt{X^2 + Y^2})}\right]. \tag{3.26}$$

To compute the velocities of the links, the derivative of (3.21) is taken and assuming that the angular speed of the input link ω_2 is known.

$$\omega_3 \times r_3 e^{i\theta_3} - \omega_4 \times r_4 e^{i\theta 4} = \omega_1 \times r_1 e^{i\theta_1} - \omega_2 \times r_2 e^{i\theta_2}. \tag{3.27}$$

All the terms in (3.27) are known with exception of ω_3 and ω_4 . Multiplying (3.27) by $-i\theta_3$ to get ω_3 and by $-i\theta_4$ to get ω_4 and rearrange the results.

$$\omega_3 = \frac{\omega_2 \times r_2 \sin(\theta_4 - \theta_2)}{r_3 \times \sin(\theta_4 - \theta_3)}, \tag{3.28}$$

$$\omega_4 = \frac{\omega_2 \times r_2 \sin(\theta_3 - \theta_2)}{r_3 \times \sin(\theta_4 - \theta_3)}. \tag{3.29}$$

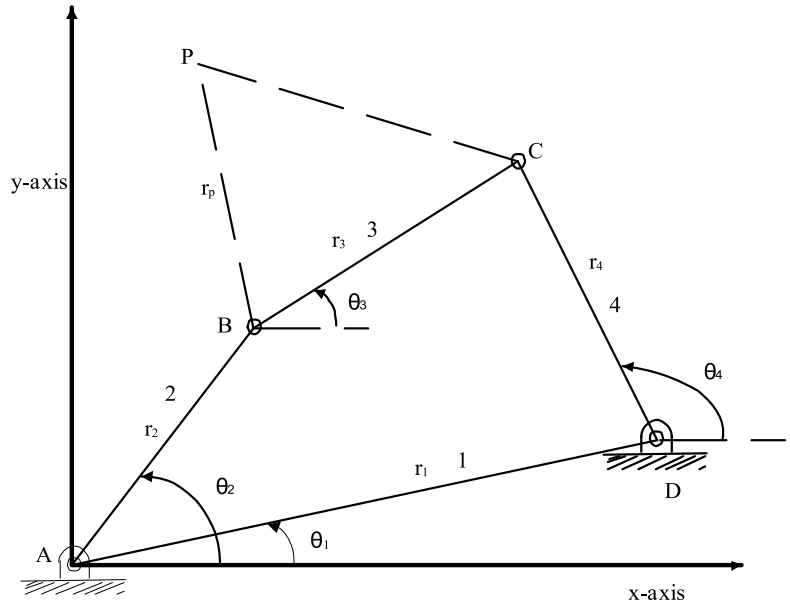


Figure 3.11: Schematic Diagram of a Four-bar Mechanism.

3.3.2.2. Linkage Synthesis and Kinematics at Constant Velocity

For a four-bar mechanism, Frendentein's method is used analytically to synthesize the link lengths [29]. It requires measuring the angular positions of the input and output links at three different positions. For the kinematic analysis of the mechanism, the input link is driven at a constant angular velocity. It is easy to determine the path of the coupler link and the output link by ploting their angular positions, velocities and accelarations over a valid range of motion. Adjustment can always be made to the links length and initial conditions to define a specific path.

3.3. Synchronization and Timing of the Multi-body System

For the various subsystems to work together as a unit achieving a common goal, synchronization and timing becomes an important part. It, mainly, deals with the computation of the positions of the objects and times required for the various actuators (e.g pneumatic cylinder) to accompolish a closed loop control actuation. Consider a network of coveyors with two boxes on them at different locations as shown in Fig. 3.12.

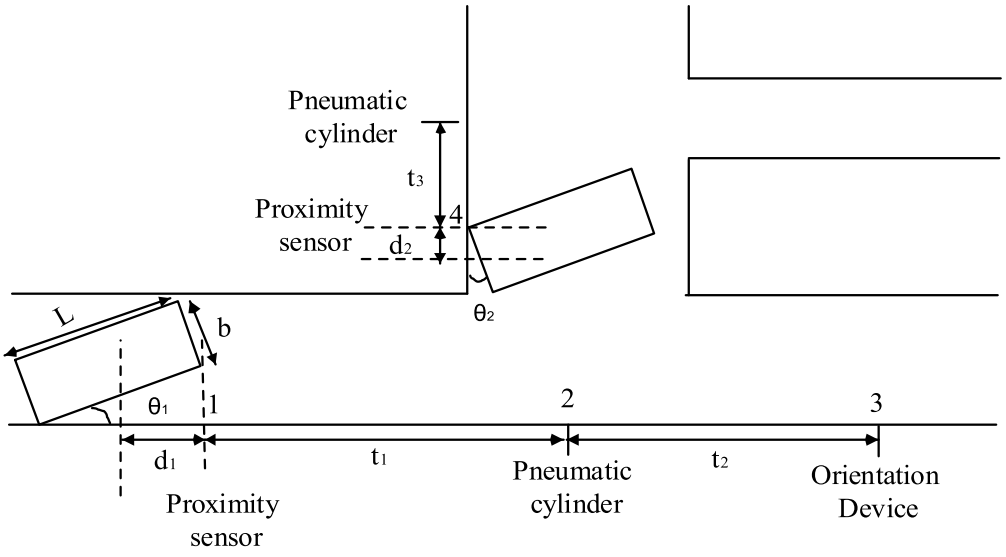


Figure 3.12: Schematic Diagram of the Objects Time Estimation on the Conveyor.

For a conveyor that travels with constant speed, time taken for an object at point 1 to reach point 2 is t_1, and from 2 to 3 is t_3. These times are constants since the respective distances and conveyor's speed are also constants. Similarly, time t_3 from point 4 to the pneumatic cylinder is constant. If the response times of the actuators are assumed to be negligible, for any given orientation of the object the time required for the actuator to meet the object at its mid-point can be computed as follows;

$$T_1 = d_1 + t_1, \tag{3.30}$$

$$T_2 = T_1 + t_1, \tag{3.31}$$

$$T_3 = d_2 + t_3. \tag{3.32}$$

where T_1 is the total timing period of pneumatic cylinder at position 2 from the time the object is detected at position 1, T_2 is the total timing period of the second pneumatic cylinder from the time the object is observed at position 4, T_3 is the total timing period of the orientation mechanism from the time the object is detected at point 1 and the Delays d_1 and d_2 are the times for half of the object to pass the sensors and are computed as follows;

$$d_1 = \frac{0.5l\cos\theta_1}{2\pi dkn} = \frac{l\cos\theta_1}{4\pi dkn}, \tag{3.33}$$

$$d_2 = \frac{0.5b\cos\theta_2}{2\pi dkn} = \frac{b\cos\theta_2}{4\pi dkn}. \tag{3.34}$$

where l and b represent the length and breadth of the object, respectivey. θ_1 and θ_2 are the inclanations of the object at two different positions.

CHAPTER FOUR

4. RESULTS AND DISCUSSIONS

4.1. Introduction

The prototype of the proposed system is depicted in Fig. 4.1. The system consists of a network of three belt conveyors. Each of the producst A, B and B with defects is routed into a seperate conveyor after being analyzed. The images of the product moving on the conveyor are captured by the camera C. Proximity sensor S1 acknowledges the presence of a passing objects. If the object passing is identified from the image processing algorithm to be product B or B with defect, pneumatic cylinder PC1 will be actuated to push the product into the conveyor 2. If the product is type A it will continue its journey to the orientation device, OD. OD is only actuated if it is found that A has a distorted orientation with respect to the x-axis. For the products pushed onto the conveyor 2, proximity sensor S2, detects their presence and if it is a product B with defect, PC2 is activated to push it into the conveyor 3 that houses defected products. Whereas, if it is normal B, it continues its journey to another terminal. Robot arm picks product A and stocks it into the palletes and since the worry orientation of A it has been handled by the OD. Pressure sensor S3 provides feedback to the OD. Proximity sensor S0 is strategically placed before the camera, and it is used to sense any object on the conveyor. The main objective of this sensor is to provide triggering signal to the camera so that snapshots are taken only when an object is passing. This eases pressure off the FPGA processor by processing frames containing objects only. The output of S0 is sent to Programmable Logic Controller (PLC) which generates the triggering signal for the camera [31]. The corresponding PLC ladder diagram is given in appendix A.

4.1. Control Flow

Figure 4.2 provides the logical control flow of the process. The critical part of the process is the image analysis which is implemented on the FPGA processor. The algorithm begins with camera initialization to acquire the images of the conveyor plane. The acquired images are preproccesed for enhancement and de-noising ahead of the analysis. The preprocessing ensures improved recognition rate and also reduces the surrounding distortion of the images

being captured. When the algorithm operated fully on the images, the results are classified according to the category they belong to. Distinguished features on the surface of the objects e.g. labels, names, logos, sizes and barcodes are used to differentiate one product from another. It should be noted that the angle θ_1, which is measured as the deviation from x-axis of product A to be corrected, depends on the minimum allowable error margin of the robot to which it can effectively pick the product. The upper limit of this angle θ_2, is limited by the extent to which the OD can align the distorted product. For square shaped product, the OD has no upper limit. It can literaly correct any angle deviation from the x-axis. With the algorithm being implemented on the FPGA, real-time processing requirement would be met [31].

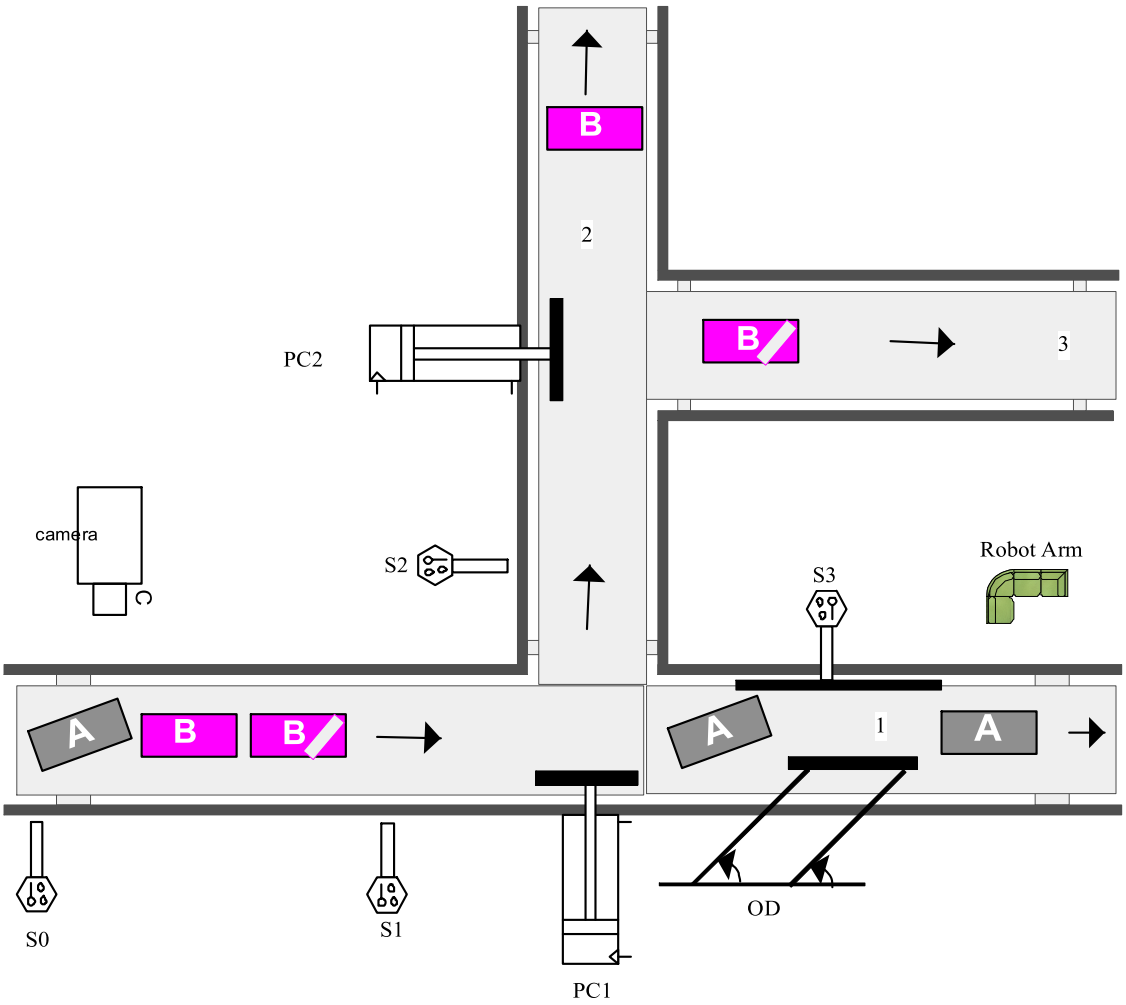


Figure 4.1: Proposed System Diagram.

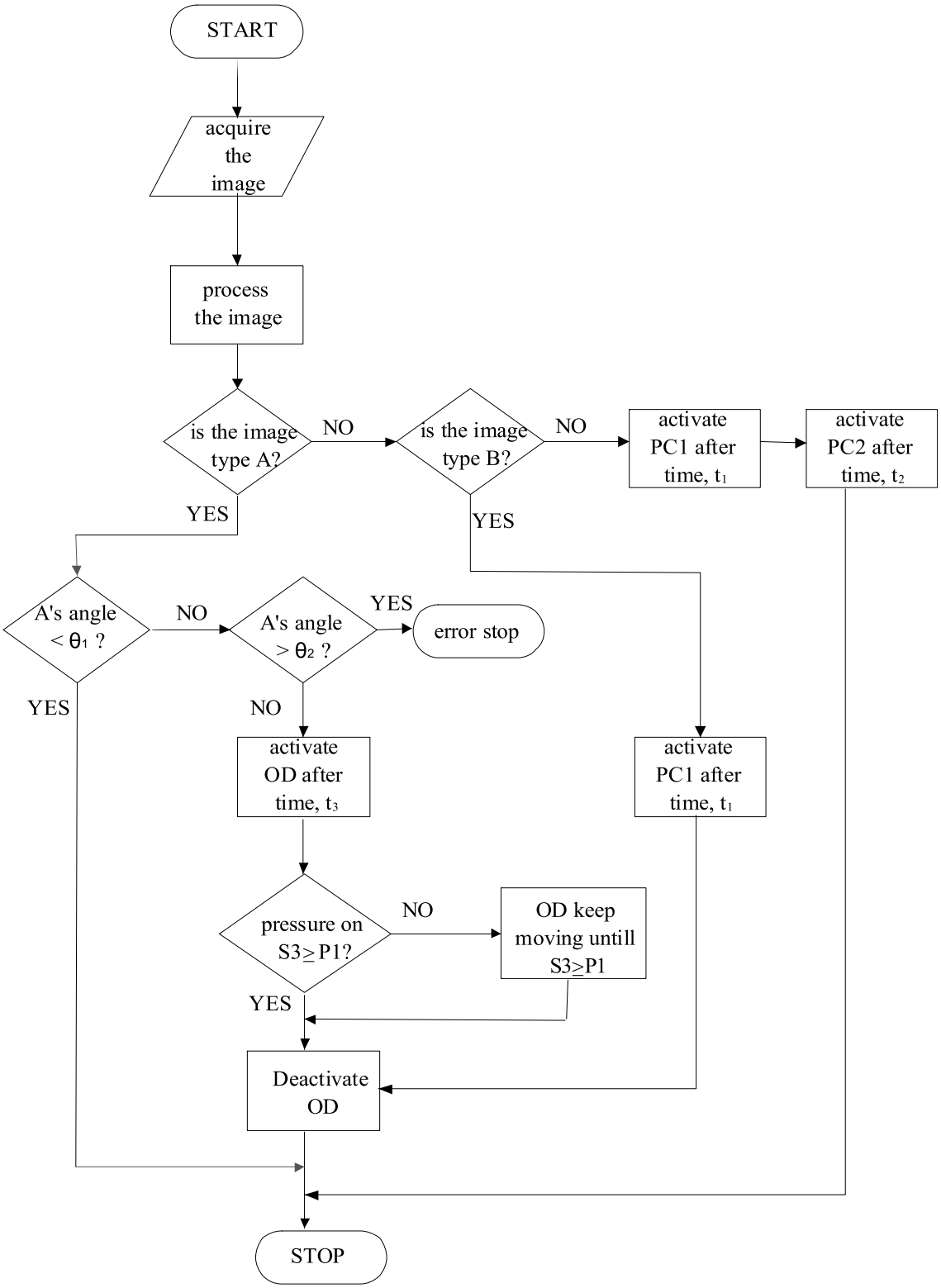


Figure 4.2: Control Flow Chart.

4.2. Image Processing Simulation Results

Initially the algorithsm were written in MATLAB programming development enviroment. Based on the codes and FPGA architecture, model-based Simulink design of the process was built. The Simulink model was adjusted and reconfigured to conform to the proposed FPGA hardware architecture. Apart from the serialization of the two dimensional image, data in the Simulink models further optimization, such as parallel allocation of the stream data for parallel processing and pipelining of the streaming data. The system blocks in the Simulink of Fig. 4.3 were modelled as subsystems beneath which other sub routine process embedded. Having evaluated by ODE45 Simulink solver and conformed to the architecture, HDL codes were automatically generated. A testbench of the simulink models created using MODELSIM software that came with Quartus ii application confirmed the workability of the model with negligible errors.

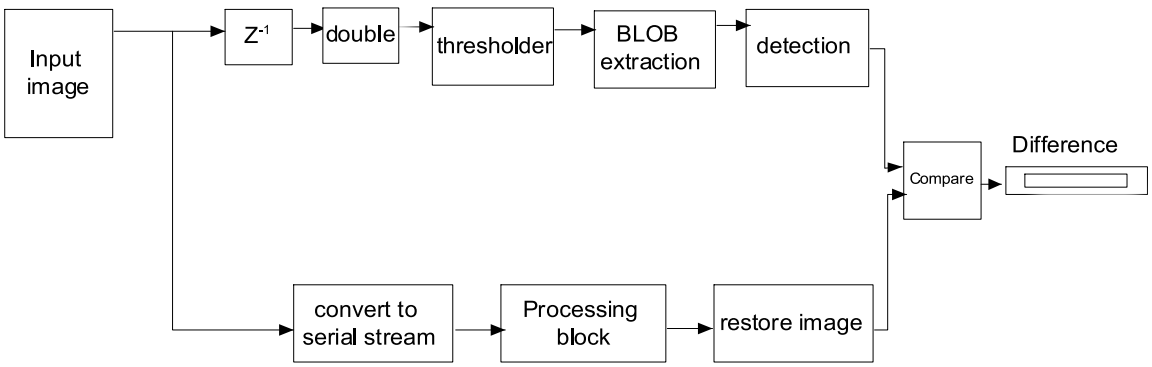


Figure 4.3: Simulink Model.

The model was targeted for low-cost Altera Cylone iii EP3C120F780 FPGA with dual on-board oscillators for generating 50MHz and 125MHz clock speed. Even at lower clock frequency of 50Mz, the resources utilization of the hardware would be very low with an extremely fast execution. Figs. 4.4-4.14 show templates of different objects stored in the image database. All images in the database are properly indexed for the purpose of identification. In Figs. 4.15-4.18, clustered images containing template 1, template 2, template 3 and template 4 were fed into the algorithm. Subsequenty, each template was correctly detected and extracted. Their exact locations in the images were determined as their inclinations with respect to the x-axis were computed as given in Table 4.1. In Fig. 4.19, an

object the same as the shape in template 1, but without mark '1' on it, is assumed to be a defective version of template 1. A clustered image containing unmarked template 1 was supplied to the system. The image was detected to be exactly as the target template 1 in the database but with no marks on it. Hence, it will be classified as a defective product. The last result was obtained when an image containing non-target object was used, and it promptly returns the detection result acknowledging non presence of a target object. For all the scenarios tested, the detection level has been good and robust to scaling and in-plane rotation of the input image. Table 4.1 provides angle correction and execution time for the software-based implementation of the algorithms on a 1.7GHz Dual Core Celeron (R) processor with 32-bit operating system. Also it shows the expected time of execution on the low-cost Altera Cylone iii EP3C120F780 FPGA processor at 50MHz clock as evaluated by ODEV45 Simulink solver.

Table 4.1: Summary of experimental results.

NO	Input Image Type	Angle Correction	ExecutionTime GPP (s)	ExecutionTime FPGA (s)	Recognition Status
1	Clustered image with Target 1	31.1^o	3.197	0.238	Target 1 found
2	Clusterd image with target 2	-1.83^o	2.849	0.301	Target 2 found
3	Clusterd image with target 3	38.16^o	2.737	0.275	Target 3 found
4	Clusterd image with target 4	54.96^o	2.564	0.212	Target 4 found
5	Clustered image with unmarked target	-22.2^o	3.337	0.222	Unmarked target found
6	Non-target clustered image	—	1.967	0.265	No target found

Figure 4.4: Template 1.

Figure 4.5: template 2.

Figure 4.6: template 3.

Figure 4.7: Template 4.

Figure 4.8: Template 5

Figure 4.9: template 6.

Figure 4.10: Template 7.

Figure 4.11: Template 8.

Figure 4.12: Template 9.

Figure 4.13: Template 10.

Figure 4.14: Template 11.

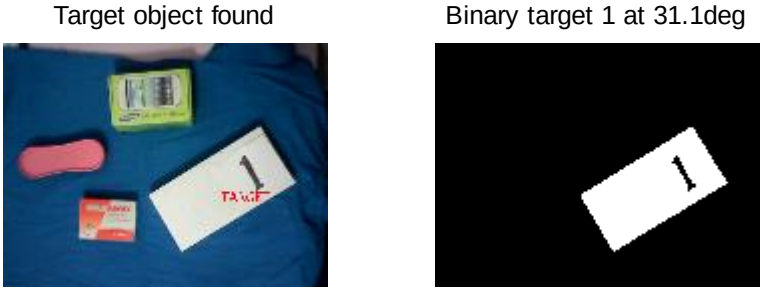


Figure 4.15: Template 1 Pre- and Post-processing results.

Figure 4.16: Template 2 Pre- and Post-processing results.

Figure 4.17: Template 3 Pre- and Post-processing results.

Figure 4.18: Template 4 Pre- and Post-processing results.

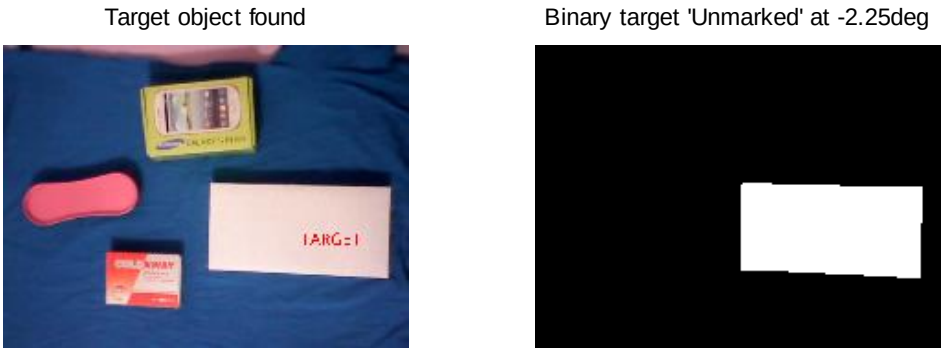


Figure 4.19: Pre- and Post-processing results of unmarked template.

Figure 4.20: Post-processing results of non-target image.

4.4. Belt Conveyor Simulation Results

According to the relevent data of the belt conveyor in Table 4.2 and the mathematical model derived in Chapter (3), the output armature e.m.f (E_a) was served as input source block of the

conveyor system model. Six gain blocks represent the motor constants as contained in the mathematical model. The constants are the induction motors' inertia, efficiency, e.m.f onstant (K_e), torque constant (K_t), rotor armature resistance (R_a) and inductance (L_a). In addition, armature current (I_a), electrical torque (T_e) and the angular velocity are modelled as the output blocks of the system model. Subsequenty, the model-based Simulink version of the system is created and simulated as shown in Fig 4.21.

Table 4.2: Induction Motor Parameters.

Parameters	Values
Efficiency	0.83
Power	1.5Kw
Armature Resistance (R_a)	10Ω
Amature Inductance (L_a)	5mH
Torque Constant (K_t)	0.09Nm/min
Back E.M.F Force Constant (K_e)	0.09V.revs/min
Inertia (J)	0.014Kgm^2
Armature Voltage (E_a)	230/400V
Frequency	50Hz

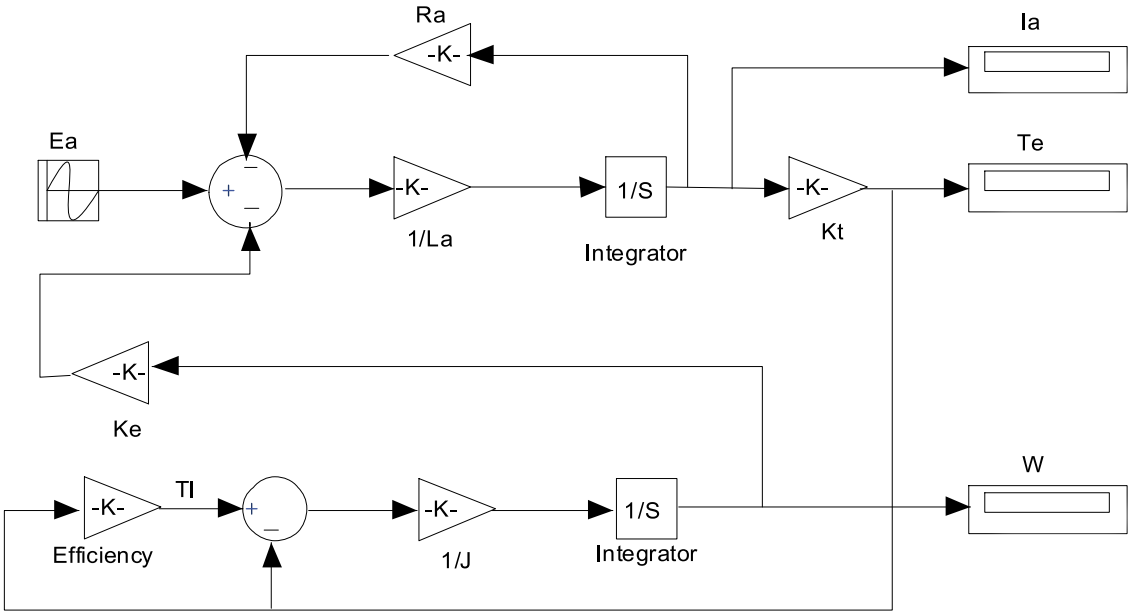


Figure 4.21: Belt conveyor simulink model.

According to the set simulation parameters of the relevent data and Simulink model of the belt conveyor, both static and dynamic behavior of the systen can be obtained. Within the scope of this work, the relationship between the belt tension, speed and capacity (load) becomes paramount. This is to ensure accurate timing and sychronization of the various units within the system. Fig. 4.22 shows how the belt tension increases as the external load or conveyor capacity is increased. It worths noting that at no-load there is tention build up and this tension is in addition to the starting tension which must be overcome by the motor's torque during start up. The additional tension is attributable to the gravitational pull due to the weight of the belt and idlers. The speed load graph of Fig. 4.23 was plotted at constant motor power of 15 kw. It indicates that the speed of the conveyor is limited by its load capacity. Higher speed can be attained at relatively low capacity. The speed rapidly drops the same as the initial motor power with increase of capacity. At about 80 kg the drop in speed becomes relatively low and steady. This represents the capacity of the belt at that point. Beyond that, the speed continues to drop but minimally. This is due to that fact that, the increased load adds up to the total tension in the belt which reduces the effective driving torque of the motor. Further increase in load pushes the conveyor to the limit where a belt slip-up is attained. At the slip-up point the speed of the belt is reduced to zero despite the rotation of the driving pulley.

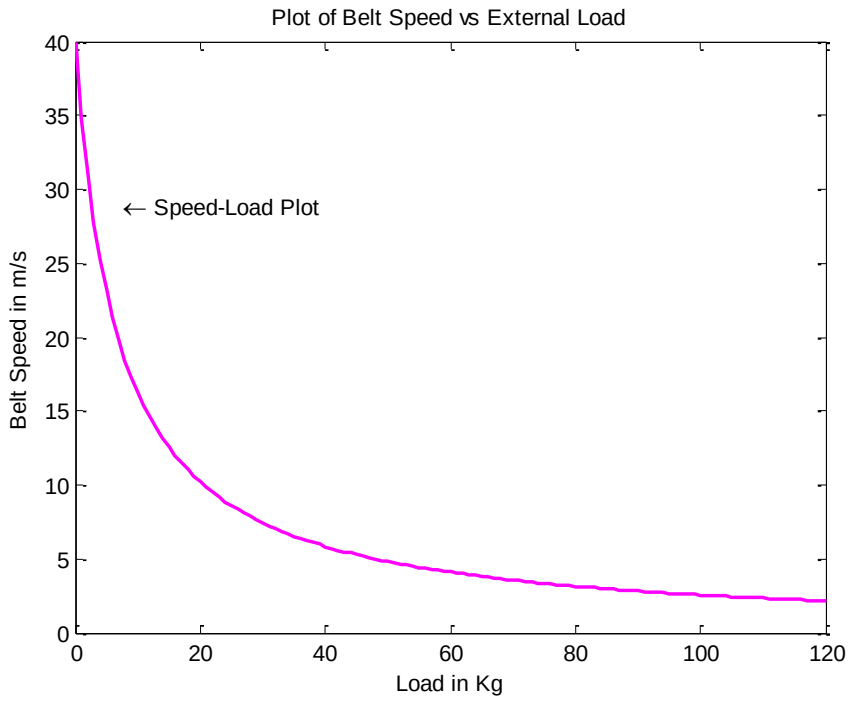


Figure 4.22: Conveyor Belt Speed vs Load.

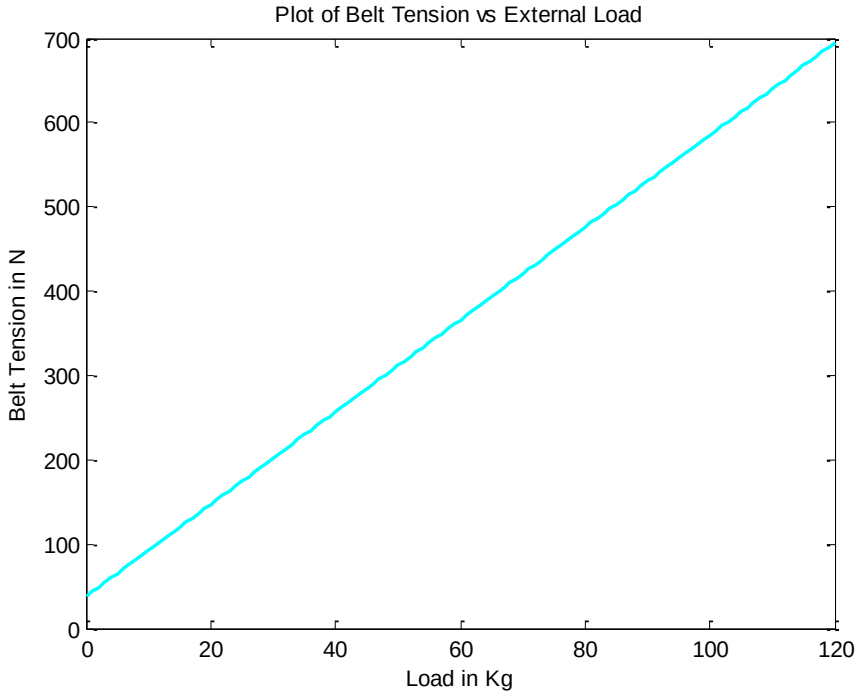


Figure 4.23: Belt Conveyor Tension vs Load.

4.5. Orientation Mechanism Simulation Results

In the OD, which is basically a four-bar mechanism, modification was made from the Grashof Singular Quadraliteral Linkage. Both the input and the output joints are driven by two similar synchronized induction motors. The trajectory of the coupler link at both ends, joint B and C, are considered. The two joints represent the displacement of the device on the conveyor plane. Figs. 4.24 and 4.25 show plots of relative displacement, velocity and accelaration against time of the two points on B and C as measured from a reference point. The reference point from which the measurement is taken and modelled is on the same axis with joint A and D of the fixed link. The origion of the reference point is placed at the end wall of the conveyor and its maximum allowable displacement is equivalent to the conveyor's width. From both plots, it worths noting that none of them is harmonic. As the length of both input and output links are made shorter relative to the coupler link, relative velocities become more sinosoidal while their relative accelarations obey cosine rule. The mechanism was simulated over a period of 2 seconds with angular speed of both the input and output links as 3.142 $rads^{-1}$.

At time $t = 0$, the relative displacement and velocities of both points are also zero while accelaration attains its maximum value. As the points begin to deaccelarate their relative velocities pick up and attain their maximum values of 3.14 ms^{-1} at exactly $t = 0.5s$. At this point, maximum displacement from the reference point is also attained and the reference point is now placed on the other wall of the conveyor. Thereafter the velocity begins to fall until it becomes again zero at $t = 1s$. This point corresponds to the point at which the displacement of the reference point complete a round trip and return to its origin. At the same time, accelaration attains another maximum but this time in the opposite direction to previously attained at $t = 0$. Between $t = 1s$ and $t = 2s$ the process repeats itself with acceleration decreasing from maximum negative value back to the maximum positive value. The displacement is now computed as a reduction from the previously accumulated value of the round trip displacement value. The only difference between Figs. 4.24 and Fig. 4.25 is the accelaration values. In Fig. 4.24 the maximum accelarations during the forward stroke is 12.29 ms^{-2} and -7.46 ms^{-2} during the reverse stroke. In Fig. 4.14, the maximum accelaration in forward and reverse strokes are 10.93 ms^{-2} and -8.82 ms^{-2} respectively. The disparity is obvious due to the fact that the origion of the output link, D, is further away from the refrence point compared to the origion of the input link, A. Due to this fact, point B must accelerate

faster in the forward stroke and slower in the reverse stroke than point C for input and output links to be synchronized.

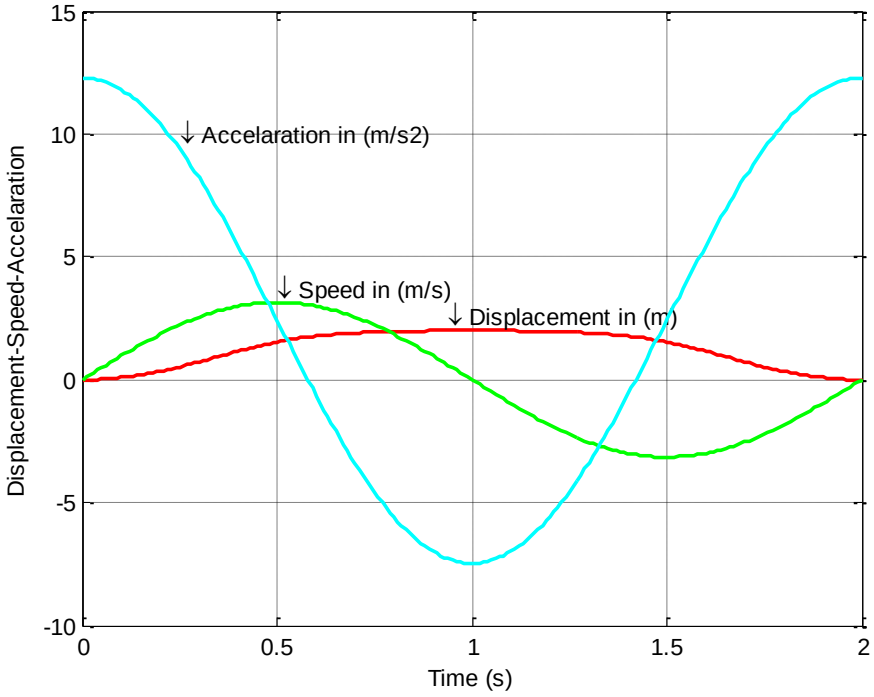


Figure 4.24: Trajectory of joint B against Time.

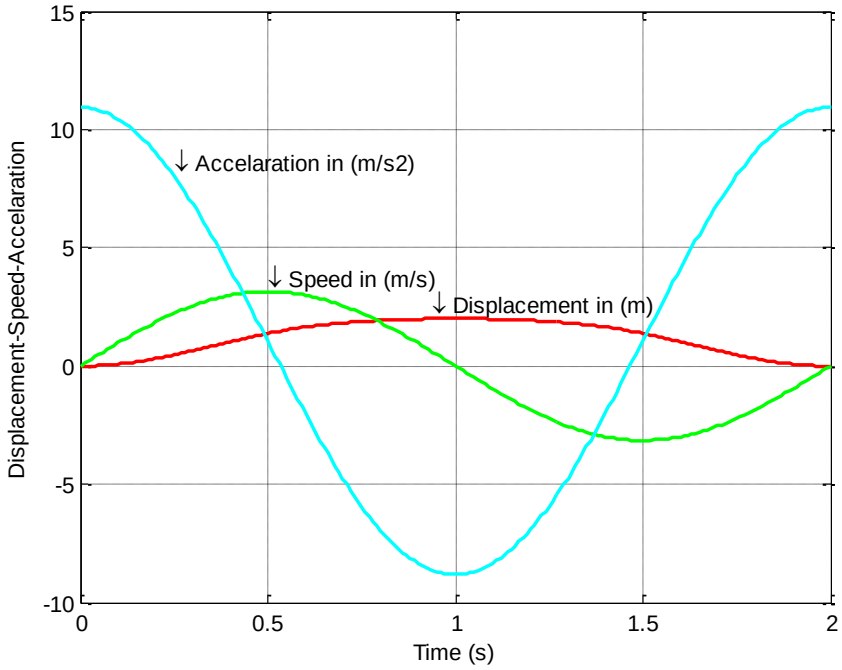


Figure 4.25: Trajectory of joint C against Time.

4.6 Discussions

The simulation results presented above have covered all the three fundamental units of the proposed system: FPGA architecture, image processing algorithms and the belt conveyor modeling. Though, it can be argued that some of the problems emanating from the GPP-based processing can, alternatively, be solved using Real-Time Operating System (RTOS) on the GPP processor. The RTOS systems come with additional cost and do not solve all the problems. Moreover, the FPGA architecture proposed has leaverages over the RTOS. For instance, in each clock cycle 4-pixel data is fed into the algorithm and a minimum of four cycles is required to form 4×4 pixel data for analysis. Each of the 4-pixel data can be processed paralelly without sharing system resources and, hence, speed up the execution time. True parallel processing is not achievable in RTOS systems.

On the other hand, the image recognition algorithms are scale-invariant and robust to in-plane rotation of the objects. As long as there is difference in the overall shapes, sizes and features between one object and another, can easily be differentiated. Though, Optical Character Recognition (OCR) algorithms may distinctively read data or barcodes on the products for identification, it is not used here because it is assumed that the product may change face while being transported and the data or barcodes may be out of sight.

The simulation results and models of both OD and the belt conveyor provide a powerful means of timing and synchronization of the entire system. Since the FPGA execution time of the algorithm is in fraction of seconds (see Table 4.1), speed of the conveyor and the OD can be adjusted to give the desired optimum actuation time.

CHAPTER FIVE

5.1. Conclusions

A system for solving Quality Control and Palletization problems was proposed in this thesis work. The various units in it were modeled and simulated. It was shown that the traditional mechanical palletizer could be made intelligent with image processing algorithms. The object recognition algorithms were shown to be consistent, scale-invariant and is not affected by in-plane rotation of the object. The FPGA proposed architecture conformed to the requirements for the implementation of the image processing algorithms. The execution time is fast enough to handle real-time processing requirements. The FPGA based architecture is much better and efficient than the GPP software-based implementation using high level programming languages.

The integration of the various parts, working as a unit, to achieve QCP applications has been demostrated. The system is shown to be able to handle both the two applications at the same time with improved efficiency and cost effectiveness.

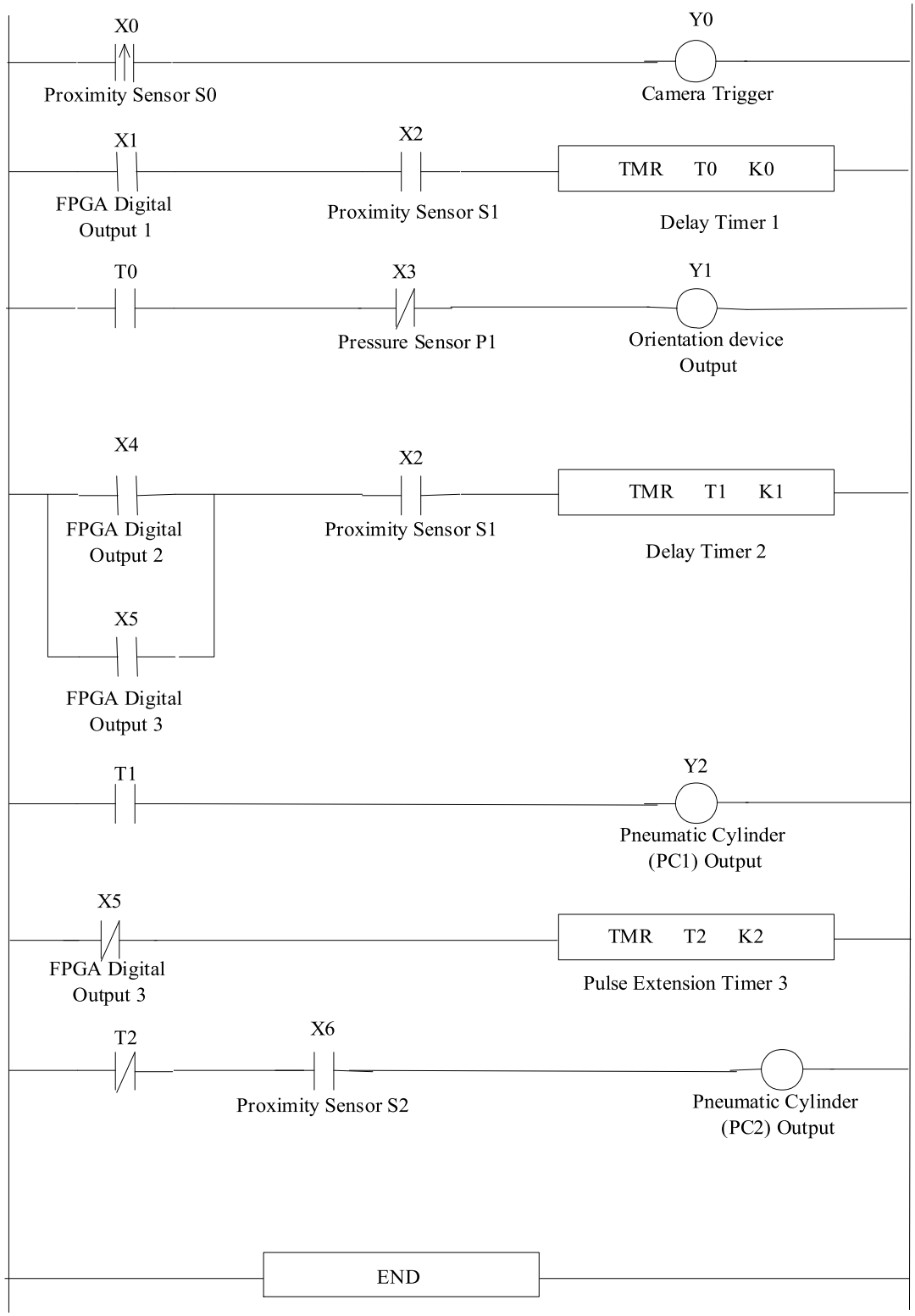
5.2. Future Work

The image recognition algorithms are able to detect target object as long as there is difference in saturation and intensity levels between the object (foreground image) and the background image. As the saturation and intensity values of the foreground and background images become similar, the detection may fail. Moreover, though the process can withstand image scaling and in-plane rotation, out-of-plane rotation of the object will lead to failure to detect the object correctly. Further research can be done to use more robust classifier such ANN and SIFT to solve the first problem. The out-of-plane rotation problem can also be solved using dedicated cascaded classifier for each possible orientation (face) of the object. In all these, the architecture of the FPGA may need to be re-designed.

The proposed mechanical palletizer can be able to correct all the angles deviation of a product with the exception of 90^o and angles very close to it. For a square-shaped product 90^o inclination does not need correction since all the sides of the product are of the same lengths and the robot will have no problem picking it. The problem becomes obvious for non square-shaped products. Future work can be done to take this problem into consideration.

APPENDIX A

PLC ladder diagram



REFERENECES

[1] C. Diederichs and S. Fatikow, "FPGA-Based Object Detection and Motion Tracking in Micro and Nano Robotics", *International Journal of Intelligent Mechatronics and Robotics*, vol. 3, no.1, 2013, pp. 27-37.

[2] A. Sultana and M. Meenakshi, "Design and Development of FPGA based Adaptive Thresholder for Image Processing Applications", *2011 IEEE Relevant Advances in Intelligent Computational Systems* (RAICS2011), 2011, pp. 633 - 637.

[3] I. Yasri, N. H. Hamid and V. V Yap, "An FPGA Implementation of Gradient Based Edge Detection Algorithm Design", *2009 IEEE International Conference on Computer Technology and Development*, 2009, pp.165-169.

[4] Z. Guo, W. Xu, Z. Chai, "Image Edge Detection Based on FPGA", *IEEE Ninth International Symposium on Distributed Computing and Applications to Business Engineering and Science*, Vol. 5, 2010, pp. 169-171.

[5] R. Harinarayan, R. Pannerselvam and M. M. Ali, "Feature Extraction of Digital Aerial Images by FPGA based implementation of edge detection algorithms", *IEEE Proceedings of International Conference on Emerging Trends in Electrical and Computer Technology* (ICETECT), 2011, pp.131-135.

[6] I. A. Qader and M. Maddix, "Real-Time Edge Detection Using TMS320C6711 DSP", *2004 IEEE Transactions on Image Processing*, vol 3, 2004, pp. 306-309.

[7] P. Dzitac and A. M Mazid, "An Efficient Control Configuration Development for High-speed Robotic Palletizing System", *IEEE Conference on Robotics, Automation and Mechatronics*, 2008 , pp. 140 - 145.

[8] H. Yu, J. Shan and X. Zhu, "Off– line Programing and Remote Control for a Palletizing Robot", *IEEE International Conference on Computer Science and Automation Engineering*, Vol. 2, 2011, pp. 58-589.

[9] J. Wu, J. Sun and W. Liu, "Design and Implementation of Video Image Edge Detection System Based on FPGA", *IEEE 3rd International Congress on Image and Signal Processing* (CISP2010), 2010, pp. 472-476.

[10] T. Shivanand, S. Rahman and G. Pillai, "Efficient and Robust Detection and Recognition of Objects in Grayscale Images", *IEEE International Conference on Computational Intelligence and Computing Research* (ICCIC2010), 2010, pp 1-6.

[11] S. N. Kalitzin, J. J. Staal and B. M. Ter Haar, "Image Segmentation and Object Recognition by Bayesian Grouping", *IEEE International Conference on Image processing*, Vol. 3, 2000, pp. 580 - 583.

[12] Z. Wang, H. Xiao, W. He and F. Wen, "Real-time SIFT-based Object Recognition System", *Proceedings of 2013 IEEE International Conference on Mechatronics and Automation*, Vol. 5, 2013, pp. 1361-1366.

[13] G. Xu, X. Wu and L. Liu, "Real-time Pedestrian Detection Based on Edge Factor and Histogram of Oriented Gradient", *Proceeding of the IEEE International Conference on Information and Automation*, Vol. 9, 2011, pp. 384-389.

[14] M. F. Talu and I. Turkoglu, "A Novel Object Recognition Method Based on Improved Edge Tracing for Binary Images", *IEEE International Conference on Application of Information and Communication Technologies* (AICT), 2009, pp. 1-5.

[15] D. Ta, W. Chen and N. Gelfand, "Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors" *IEEE Conference on Computer Vision and Pattern Recognition*, (CVPR2009), 2009 , pp. 2937-2944.

[16] G. Orchard, J. G. Martin and R. J. Vogelstein, "Fast Neuromimetic Object Recognition Using FPGA Outperforms GPU Implementations", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 24, no. 8, 2013, pp. 1239-1252.

[17] G. Anusha1, T. J Prasad and D. S Narayana, "Implementation of SOBEL Edge Detection on FPGA", *International Journal of Computer Trends and Technology*, Vol. 3(3), 2012, pp. 472-475.

[18] C. Topal, C. Akinlar and Y. Genc, "Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection", *IEEE International Conference on Pattern Recognition*, 2010, pp. 2424-2427.

[19] G. Xu, X. Wu and L. Liu, "Real-time Pedestrian Detection Based on Edge Factor and Histogram of Oriented Gradient", *Proceeding of the IEEE International Conference on Information and Automation*, 2011, pp. 384-389.

[20] L. Braun, D. Gohringer and T.Perschke, "Adaptive Real-time Image Processing Exploiting two Dimensional Reconfigurable Architecture", *Journal of Real-time Image Processing*, Vol. 4, 2008, pp. 105-125.

[21] S. Milan, H. Vaclav, and B. Roger, 2008. *Image Processing, Analysis and Machine Vision*. Thompson. Toronto, USA.

[22] F. Smach, M. Atri and J. Miteran, "Design of a Neural Networks Classifier for Face Detection", *Journal of Computer Science*, Vol. 2, no. 3, 2006, pp. 257-260.

[23] K. Kim, S. Lee and J. Young Kim, "A Configurable Heterogeneous Multicore Architecture with Cellular Neural Network for Real-time Object Recognition", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 19, no. 11, 2009, pp. 1612-1622.

[24] Q. Lin, Y. Han and H. Hahn, "Real-time Lane Detection Based on Extended Edge linking Algorithm", *Second International Conference on Computer Research and Development*, Vol. 9, 2010, pp.725-730.

[25] C. Topal, O. Ozsen and C. Akinlar, "Real-time Edge Segment Detection with Edge Drawing Algorithm", *7th International Symposium on Image and Signal Processing and Analysis* (ISPA), 2011, pp. 313-318.

[26] B. Alexander, H. Herpers and B. K. Kenneth, "Hardware Accelaration of BLOB Detection for Image Processing", *Third Internatinal Conference on Advances in Circuits, Electronics and Micro-electronics*, 2010, pp. 28-33.

[27] M. Pearsone, M. Nibouche and A.G. Pipe, "A Biologically Inspired FPGA-based Implementation of a Tactile Sensory System for Object Recognition and Texture Discrimination", *IEEE International Conference on Field Programmable Logic and Applications*, 2006, pp. 1-4.

[28] K. Kim, J. Kim and S. Kang, "Object Recognition for Cell Manufacturing System", *IEEE 9th International Conference on Ubiquitous Robots and Ambient Intelligent*, (URAI2012), 2012, pp. 512-514.

[29] T. Kuo-Hsiung, C. Ruey-Fong, C. Juei-Long and S. Te-Wei, " Modeling and Analysis of Belt Conveyor using Bond Graph Approach", *6th IEEE Conference on Industrial Electronics and Applications* (ICIEA), 2011, pp. 2717-2722.

[30] Y. Chen and H. Xue, "Model and Dynamic Simulation of Belt Conveyor", *International Conference on Intelligent System Design and Engineering Applications*, 2010, pp. 949-951.

[31] A. M. Ashir, A. A. Ata and M. S. Salman, "FPGA-Based Image Processing System for Quality Control and Palletization Applications", *2014 IEEE International Conference on Autonomous Robot Systems and Competitions* (ICARSC), Espinho, Protuguese, 2014, pp. 285-290.

[32] H. Nakazawa , M. Ishida and K. Sawada , "Multimodal Bio-Image Sensor for Real-Time Proton and Fluorescence Imaging", *IEEE International Conference Publications on Solid-state Sensors, Actuators and Microsystems*, 2011, pp. 1966-1969.

[33] H. Abe, "Device Technologies for High Quality and Smaller Pixel in CCD and CMOS Image Sensors", *IEEE International Conference on International Electron Devices Meeting* (IEDM2004), 2004, pp. 989-992.

[34] K. Sawada, "CCD- based Ion Image Sensors for Novel Bio-Imaging Fusion of Sensor Technology and LSI Technology", *IEEE International Conference on Optical MEMS and Nanophotonics* (OMN2012), 2012, pp.214-215.

[35] B. S. Carlson, "Comparison of Modern CCD and CMOS Image Sensor Technologies and Systems for Low Resolution Imaging", *IEEE Proceedings on Sensors*, 2002, Vol. 1, pp.171-176.

[36] D. L. Losee, J. A. Johnson, S. L. Kosman, N. T. Kurfiss and G. G. Putnam, "All-ITO Gate, Two-Phase CCD Image Sensor Technology" *IEEE International Electron Devices Meeting* (IEDM 2003), 2003, pp.1641-1644.