An Evolvable Hardware System for Automatic Optical Inspection

Jonathan Evans

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
March 2004

Abstract

The use of Automatic Optical Inspection systems is very important for the manufacture of Printed Circuit Boards. This type of system is critical for quality control in both low volume and high volume manufacture where companies invest heavily in automated systems to achieve consistent detection of faults. These systems are real-time systems and are often embedded. The systems are usually integrated into the manufacturing process and report faults for analysis by process control systems. The systems give improved performance over human inspection with higher consistency of fault detection.

This thesis investigates the use of System-On-Chip technology as the basis for a low cost Automatic Optical Inspection system. Novel object recognition and image registration algorithms are developed and targeted for a System-On-Chip platform. Execution time analysis of the novel algorithms is performed. This analysis shows that the application's performance criteria can be met through enhancing the platform using a Digital Signal Processor.

Fast and accurate object recognition is an important class of algorithms for inspection systems. This thesis shows that techniques based on the Hough Transform are too computationally expensive for the recognition of Integrated Circuits. The thesis presents a novel low complexity technique based on Region Growing which is robust and efficient for complex images.

Effective and efficient image registration is a fundamental task in inspection systems. This thesis presents a new approach to detecting placement errors of Integrated Circuits. The approach is based on the use of a Genetic Algorithm to derive transformations for matching a captured image to a reference image.

This thesis presents execution time studies of the novel algorithms on an enhanced target platform. The image registration system is partitioned across the enhanced platform. The Digital Signal Processor executes the fitness function of the Genetic Algorithm. The rest of the algorithm runs on the System-On-Chip platform.

The techniques developed in this work can be used to detect further types of faults on test samples such as placement errors of resistors and capacitors. The work therefore forms the basis of a high functionality low cost Automatic Optical Inspection system.

Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the School of Engineering and Electronics at The University of Edinburgh.

Jonathan R. Evans

Acknowledgements

Thanks to my Supervisor Dr Tughrul Arslan for his advice and support throughout this project.
Thanks to my second supervisor Dr David Renshaw for his suggestions.

Thanks to Robert Thompson, Dr Peter Hillman and Dr Andrew Peacock for their suggestions.

Thanks to Integral Vision, Quantum Electronics, Matrix Technologies and Zot Engineering.

Thanks to Dr Rob Nayler and Dr Tim Reynoldson at the Department of Trade and Industry.

Contents

List of figures

List of tables

Acronyms and abbreviations

ADS	ARM Developer Suite
AOI	Automatic Optical Inspection
API	Application Programmer Interface
ASIP	Application Specific Instruction Set Processors
ASIC	Application Specific Integrated Circuits
CCS	Code Composer Studio
CPF	Cumulative Probability Function
CPS	Curvature Primal Sketch
DCT	Discrete Cosine Transform
DMA	Direct Memory Access
DSP	Digital Signal Processor
EHW	Evolvable Hardware
FPGA	Field Programmable Gate Array
GAs	Genetic Algorithms
GUI	Graphical User Interface
IC	Integrated Circuit
IDE	Integrated Development Environment
PCB	Printed Circuit Board
PGA	Parallel Genetic Algorithm
PLD	Programmable Logic Device
ROI	Region Of Interest
RTOS	Real-Time Operating System
SOC	System-On-Chip
SLS	Smoothed Local Symmetries
UML	Unified Modelling Language
VCC	Virtual Component Co-design
VM	Virtual Machine
VLSI	Very Large Scale Integration

Chapter 1
Introduction

1.1 Introduction

Automatic Optical Inspection (AOI) Systems aim to improve the accuracy and consistency of testing Printed Circuit Boards (PCBs) [4], [5] and [6]. AOI systems can detect many component types on a PCB including correct presence, displacement and rotation of Integrated Circuits (ICs), Resistors, Capacitors and Connectors. These parameters are often checked without the assistance of CAD data. CAD data specifies which components are on the board and their positions.

AOI systems are used for both low and high volume manufacture and are often integrated into a complete automated production process. These machines are of critical importance to PCB manufacturers as quality control is a high priority. AOI systems are highly complex multi-camera real-time machines and are usually embedded devices [7], [8], [9], [10] and [11]. AOI machines also have advanced user interfaces for controlling testing of samples because they are often used by operators without a programming background [12]. For these reasons the cost of AOI systems is high, making them unattractive for small to medium size companies.

The use of AOI has significant advantages over human inspection [4], [5] and [6]. Research and development into AOI systems has occurred since the 1970s and there are many machines on the market [13], [14], [15], [16] and [17].

This thesis investigates the use of System-On-Chip (SOC) technology [18] and advanced image processing techniques to produce a low cost embedded real-time AOI system based on Evolvable Hardware (EHW) [19]. To reduce system cost and system complexity the system uses a single camera.

The system does not rely on the availability of CAD data for a board. Accurate CAD data may not be present in many testing environments where there is a quick board modification and then test turn-around cycle.

The use of SOC technology has significant cost advantages over a PC Pentium based system when the system comes to high volume manufacture [18]. Through the use of SOC technology the tight integration of a micro-controller and a specialised Digital Signal Processor (DSP) will also give performance benefits. The other benefits of SOC technology such as power and area requirements are not critical for this application as AOI machines are not portable or battery powered, or very restricted in area requirements.

In this work a prototyping methodology was followed [7]. Prototyping methodologies can be divided into two main types. In 'Throw-away' prototyping involves developing a prototype to understand the system requirements. In evolutionary prototyping, a prototype evolves through several versions to the final system. This second approach was followed in this work to produce prototypes for assessment in terms of their diagnostic accuracy and their efficiency with reference to the performance requirements of an AOI system. As memory is restricted on the target system, the code size was also an issue along with the dynamic memory requirements of the code and the memory required to store images of PCBs.

The remainder of this chapter is structured as follows: in Section 1.2 the motivation for this work is presented; in Section 1.3 the main contributions of the thesis are presented; in Section 1.4 the details of the published work from this thesis is presented; in Section 1.5 the components of the complete prototype AOI system are detailed; in Section 1.6 the structure of the thesis is presented; and Section 1.7 presents a summary of this chapter.

1.2 Motivations

A major motivation of this work is to develop a new approach to the development of AOI machines through the use of EHW. The EHW is made up of standard programmable cores, an SOC and a high performance DSP. Although circuits are not evolved to configure Programmable Logic Devices (PLDs) or Field Programmable Gate Arrays (FPGAs) as in [19], a Genetic Algorithm (GA) is employed to solve an image registration problem [20]. The system is an EHW system because a customised hardware platform is combined with an adaptive registration mechanism in software.

This work is also proof of concept exercise for a low cost AOI system using EHW based on SOC technology. The work develops image processing functions for generating diagnostics. The software techniques are optimised and partitioned across an ARM7 SOC and Texas EVM6201

DSP. The motivation is to take the results of running the software and produce a customised SOC implementation which will meet the performance requirements of the application. A motivation is therefore to produce a library of image processing functions optimised for embedded processors. To achieve this the software uses a vision library which is written in C++ which is described in [21]. The library contains code for edge detection, segmentation techniques such as region growing and morphological operators. These techniques are used by the object recognition and image registration code developed for this thesis. To integrate with the library the software for this work was also developed in C [22] and C++ [23].

In this work the time required to process a PCB test sample is in the range of 1 to 2 minutes as specified through verbal communication with Zot Engineering [24]. In this work highly complex industrial test samples are processed. The samples contain features such as Integrated Circuits, Resistors, Capacitors, Connectors, Identifiers and Tracks. Due to this complexity severe demands are made on the software architecture and the hardware platform on which they execute. Given these demands, an important motivation is to show that through targeting the software for high performance programmable cores it can be predicted that the computational demands of the application can be met.

An important class of algorithms for inspection systems are object recognition techniques. Existing methods of object recognition such as the Hough Transform are computationally complex [1] [25]. In this thesis the motivation is to produce object recognition techniques which have significant reduced complexity over the Hough Transform [26].

Another important class of algorithms for inspection systems are image registration methods. Correlation techniques are well known techniques for detecting alignment and rotation of components in inspection systems [27]. However for the type of problem where there are arbitrary rotations and displacements of ICs these techniques are inefficient [25]. GAs have been found to be fast and efficient search procedures for solving complex optimisation problems [28] [29] [30] [31] [32] . Therefore in this thesis a further motivation is to develop fast and efficient image registration code based on a GA.

The overall cost of the AOI system developed for this thesis is a significant motivating factor as the aim is to design a low cost system. A final motivation is therefore for the image processing techniques to work with a single camera system. The use of a single camera significantly reduces the complexity of the software. This reduces the system's hardware platform require-

ments and the overall cost of the complete AOI system.

1.3 Contributions

The main contribution of this thesis is to prove the concept of a low cost AOI system based on EHW using SOC technology.

In the work advanced image processing techniques are developed. These techniques include novel object recognition and image registration methods applied to the problem of detecting presence, alignment and displacement of ICs. These software techniques are partitioned, optimised and then targeted for programmable cores, an ARM7 SOC with an advanced DSP enhancement. Detailed performance studies of the execution time of the software are then given. A major contribution of the work is to show that through these studies it can be predicted that the demands of the application can be met through an enhanced platform.

The work forms the basis of a high functionality AOI system which will give accurate diagnostics for PCBs in both low and high volume manufacture. The work can be taken further to produce an integrated SOC which combines a micro-controller and an advanced DSP which will give further benefits in performance and cost.

1.4 Publications

The work presented in this thesis has been published in a journal and two major conferences.

1.4.1 Refereed Journal

J. Evans and T. Arslan, Enhanced Image detection on an ARM based Embedded System, Design Automation for Embedded Systems: Special Issue on Embedded System Design in the UK, Kluwer Academic Publishers, July 2002 , Volume 6 , Issue 4 , pp477-487

1.4.2 Refereed Conferences

J. Evans and T. Arslan, The Implementation of an Evolvable Hardware System for Real Time Image Registration on a System-on-Chip Platform, NASA/DOD Conference on Evolvable

Hardware (EH2002), p 142-146, 2002.

J. Evans and T. Arslan, Implementation of a Robust Image Registration Algorithm on an ARM System-on-chip Platform, IEEE2002 International Symposium on Circuits and Systems (ISCAS2002) , Volume (2) p 269-272, 2002.

1.5 The Prototype AOI System

Figure 1.1 gives a system diagram for the final prototype AOI system. The diagram shows the main components of the system which are an ARM7 core module processor on the Integrator platform; the DSP integrated into an Intel based PC; the frame grabber for capturing and storing images from the camera also integrated into the PC; a VDU for reporting errors to the operator; an alignment grid for placing the sample under test; an industrial camera for capturing the images; and lighting to give sufficient illumination to the sample - effective illumination is critical to the success of the image processing. This is most important in the object recognition work in Chapter 5 where controlled consistent lighting conditions are necessary. This is because of the use of multi thresholding which segments a greyscale image according to a fixed band of intensities.

Figure 1.2 is a photograph of the test system set-up which shows a PCB test sample on the alignment grid, and a camera above it for capturing images and lighting. It should be noted that the light sources have tracing paper over the bulbs to give a more diffuse illumination across the test sample.

Figure 1.3 is a photograph of the embedded ARM Integrator platform, with an ARM7 core module and a Multi-ice debug unit for interfacing to a PC.

Specific hardware components for this prototype are as follows:-

ARM Integrator AP ASIC platform - part no KPI-0109AK
ARM Integrator CM7TDMI - part no KPI-0110AK
ARM Multi-ice debug unit - part no KPI-0066A
Texas TMS320C6201 EVM Kit
Colour video camera - part no FUM-982H
Verifocal Lens - part no T271816C
Picolo PCI frame grabber with Multicam Driver - part no 1155

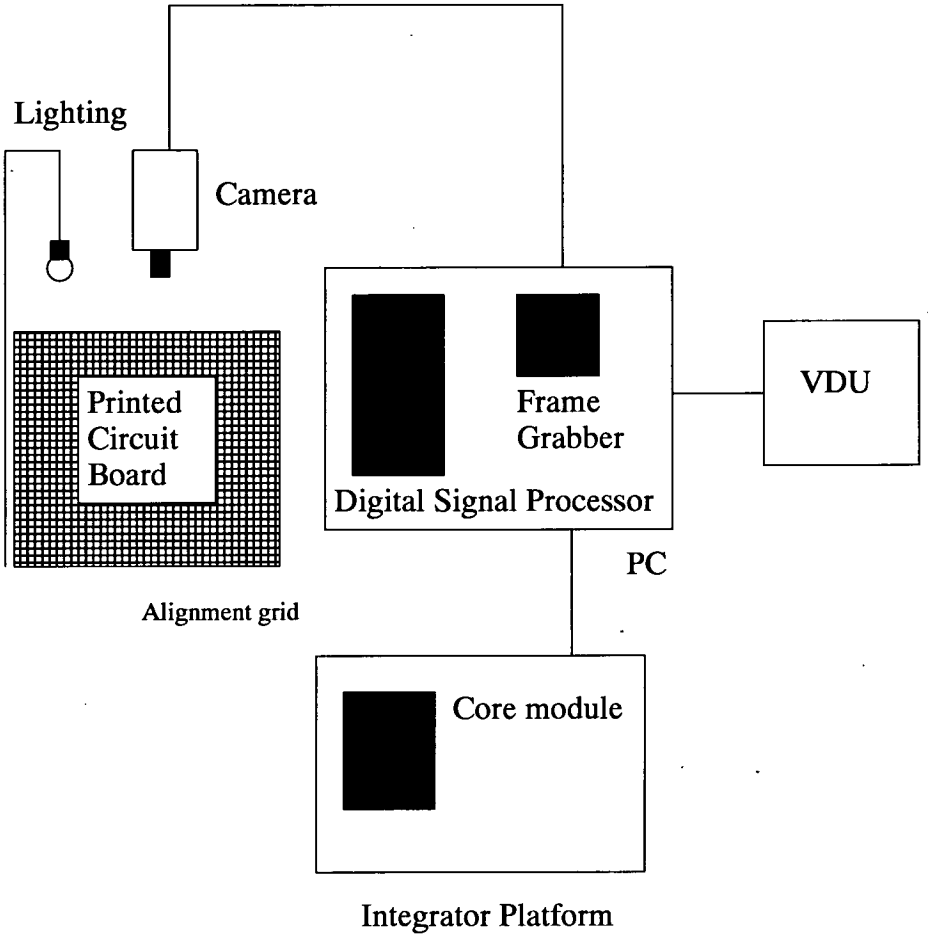

Figure 1.1: *System Diagram*

Kaiser fototechnik copy stand - part no RS 3XA(5311)
Photo flood copy stand lights - part no 5350

The resolution of the camera is 752(H) by 582(V). For all test samples acquired the distance from the camera to the object was 50cm. At this distance the number of pixels per square centimeter of the test sample is 62(H) x 66(V) pixels. The number of bits to define each greyscale pixel in an image is 8.

In this work three test samples are used. Each test sample is complex with multiple features corresponding to components such as Integrated Circuits, Resistor, Capacitors, and Connectors.

There are also other features such as IDs and tracks. The test samples used are industrial samples supplied by Zot Engineering [24], one of the industrial partners for the project.

PC based Integrated Development Environments (IDEs) used for the prototype development are: Arm Developer Suite (ADS for SOC platform) - version 1.1; and Code Composer Studio (CCS for DSP platform) - version 2.10.

The software to capture the images from the camera was Easygrab for Multicam - version 3.5.1.2.

1.6 Structure

The structure of this thesis is as follows:

Chapter 2 gives an introduction to algorithms relevant to this thesis including object recognition, image registration and GAs. The Chapter also introduces the field of EHW.

Chapter 3 gives an overview of the field of real-time and embedded systems, gives a definition of SOC technology, and describes SOC methodologies and existing work on SOC for vision systems.

Chapter 4 gives details of the development of a Hough Transform for object recognition of ICs. A reduced complexity technique based on the Hough Transform is described. The complexity of the object recognition algorithms is analysed.

Chapter 5 describes the use of region growing for object recognition, specifically applied to the problem of detection of ICs. The algorithms are targeted for an ARM7 SOC. The complexity of the algorithms are analysed and compared to the Hough Transforms.

Chapter 6 describes the use of a GA for image registration. The main stages in the optimisation of the algorithms are described. The performance of the algorithms are analysed against the performance requirements of the application. To meet the performance requirements partitioning of the registration system across an SOC and a DSP is carried out.

Chapter 7 presents conclusions and a summary of the thesis discussing limitations and suggestions of topics for future research.

1.7 Summary

The theme of this thesis is to use EHW based on SOC technology and advanced image processing techniques for an AOI system. The motivation is to prove the concept of a low cost AOI system based on EHW for inspection of PCBs.

The thesis makes a number of contributions to the field of image processing including object recognition and image registration techniques for inspection systems. The thesis shows that through implementing the techniques, optimising them and partitioning them on an SOC platform with a DSP enhancement it can be predicted that the performance criteria of the system can be met. The thesis therefore also makes a contribution to the field of embedded real-time systems and EHW utilising SOC technology.

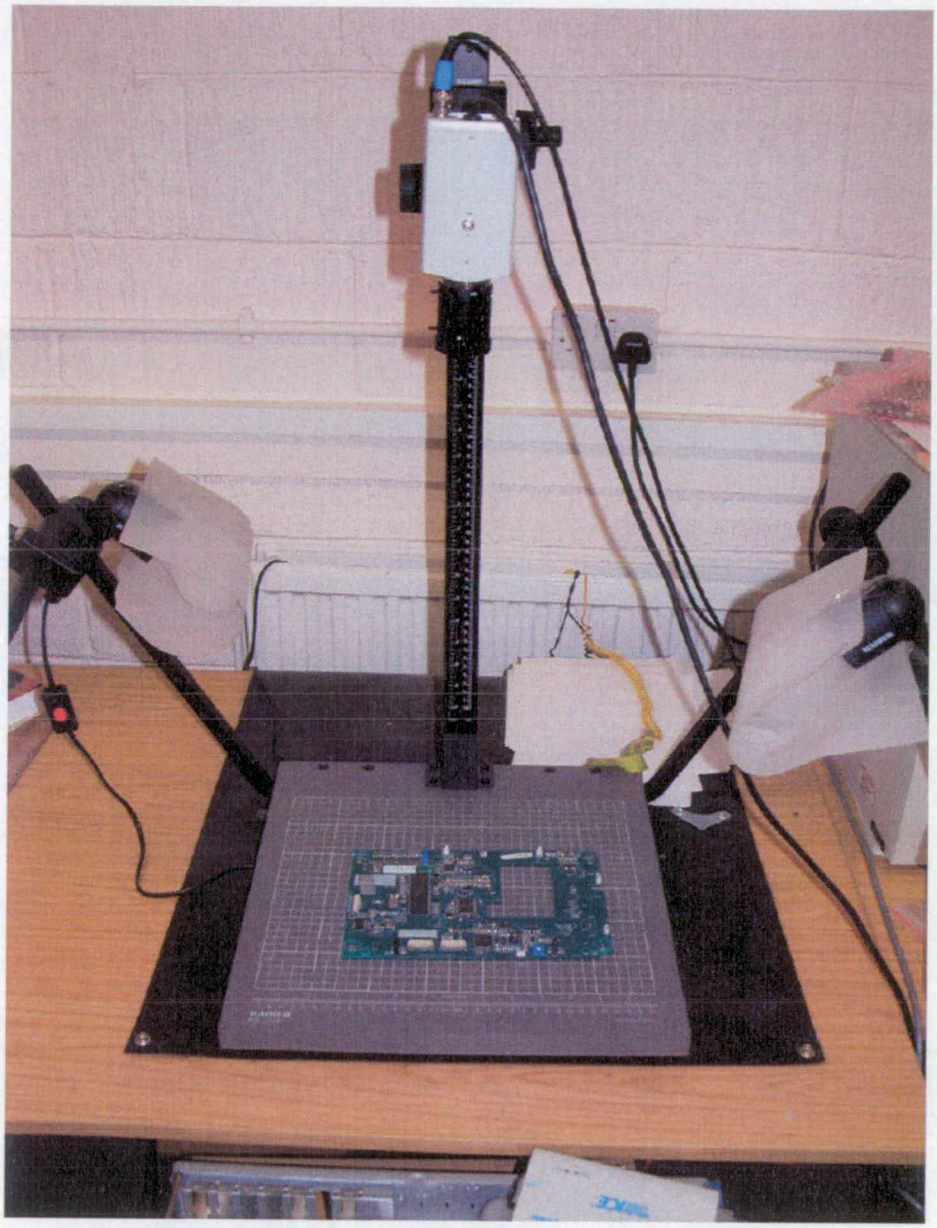

Figure 1.2: *Test Set-up Photograph*

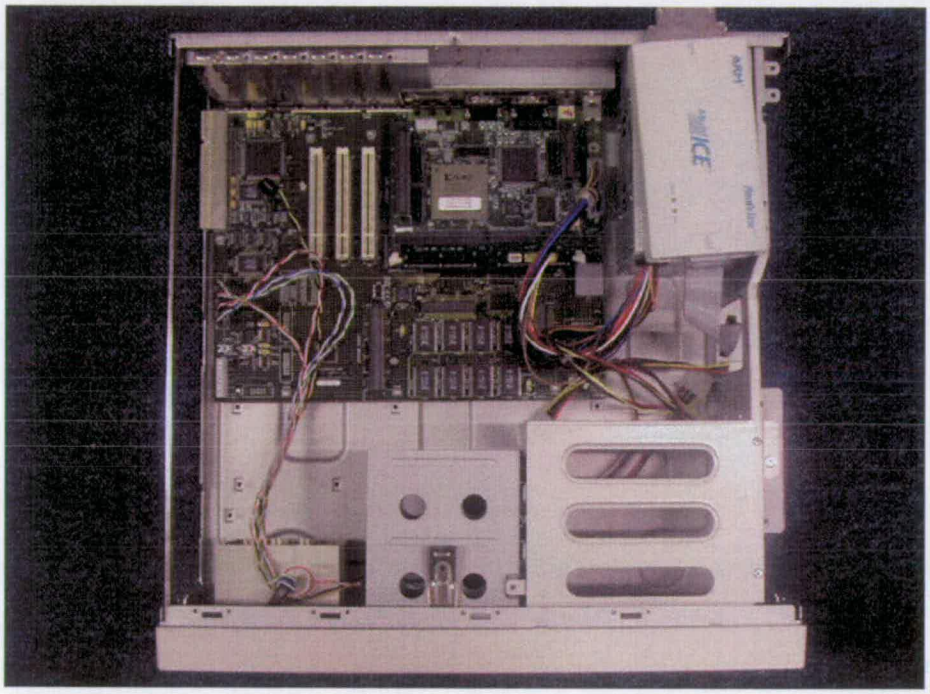

Figure 1.3: *Embedded Platform Photograph*

Chapter 2
Algorithms for Automatic Optical Inspection

2.1 Introduction

This Chapter introduces the machine vision algorithms and evolutionary techniques used in AOI systems. Machine vision edge detection operators are essential to the analysis of images of PCBs. In this work two edge detection operators are used. The main operator used for the object recognition and the image registration process is the SUSAN operator which is described in [33]. An alternative operator is the Sobel operator [1] and it is used for the probability processing technique for enhanced object recognition, Section 5.5.2.

This Chapter describes the importance of segmentation in machine vision and its importance in this work. The Hough Transform and Region Growing segmentation techniques are detailed.

The object recognition tasks solved in this thesis is a 2D recognition task. This Chapter describes existing 2D shape representation and recognition techniques.

This Chapter goes on to describe image registration methods, which are important techniques for inspection systems. This work uses a registration method based on a genetic search so the principles of Genetic Algorithms (GAs) are given. This description includes a basic algorithm and important genetic operators used such as crossover, mutation, elitism and hill climbing. This Chapter then describes other works which use a GA for image registration.

This Chapter discusses the suitability of the use of GAs for a Real-time embedded system.

This Chapter gives an introduction to the field of EHW and defines how the work in this thesis differs from the majority of work in the area.

Finally, this Chapter describes existing work in AOI systems for PCB analysis in published papers and patents.

This Chapter proceeds as follows: in Section 2.2 describes the SUSAN and Sobel edge de-

tection operators; in Section 2.3 discusses segmentation - the Hough Transform and Region Growing techniques; in Section 2.4 2D shape representation methods are detailed focusing on Smooth Local Symmetries; in Section 2.5 existing published research in 2D object recognition is reviewed; in Section 2.6 introduces registration methods and discusses how they are relevant to this work; in Section 2.7 discusses the principles of GAs; in Section 2.8 other works which apply GAs to image registration are detailed; in Section 2.9 the suitability of the use of GAs for embedded systems is discussed; in Section 2.10 gives an introduction to EHW; in Section 2.11 existing literature on AOI systems for PCB analysis is detailed; and Section 2.12 draws some conclusions from the chapter.

2.2 Edge Detection

Edge detection operators aim to preserve useful structural information about object boundaries. Edge detection methods find changes in intensity in an image. Edge detection is essential to most machine vision systems because changes in intensity in an image mark the boundaries of objects within a scene. Edge detection algorithms in general are discussed in [1], [25], and [27]. The edge detection is vital in the object recognition work because without accurate edge detection the recognition procedure will fail. Edge detection is important for the image registration procedure because it cuts down on the number of points to be processed.

Edge detectors can be classified into two broad classes: gradient operators and second derivative operators. Gradient operators respond with a broad peak at an edge location. The problem with gradient operators is they are sensitive to local noise. Second derivative operators respond with a zero-crossing at an edge location. They are more stable because a larger pixel neighbourhood is taken into consideration.

The edge detection operators used in this work are the SUSAN and Sobel operators. An alternative operator to these techniques is the Canny operator [34]. The Canny operator is complex, the main stages in its operation can be described as follows:-

1) Convolve the image with a Guassian operator.

2) Estimate local edge normal direction.

3) Find location of edges

4) Compute magnitude of the edge

5) Threshold edges with histeresis to eliminate spurious responses.

6) Repeat stages 1 to 5 with ascending values of Standard Deviation.

7) Aggregate final information about edges at multiple scales.

The Canny scheme aims to maximise the signal-noise ratio of an edge giving correctness in detection. The scheme also aims to minimise the distance between the true edge and the detected edge, giving accuracy of edge locations. Finally, the technique aims to minimise the responses associated with an intensity change, aiming for a single response.

The SUSAN operator is a new approach to edge detection which does not use derivatives and does not require noise reduction as defined in [33]. The Sobel operator is a gradient operator. These operators are efficient when compared to other techniques such as the Canny operator. These operators are therefore suitable for a real-time system such as the system proposed in this work. The operators are also available in the vision library used for this project [21] and therefore system development time is reduced.

2.2.1 The SUSAN Operator

The main edge detection operator used in this work is based on the work on the SUSAN operator defined in [33]. The SUSAN approach carries out edge detection (one dimensional feature detection) and 'corner' detection (two dimensional feature detection, including corners and junctions) and structure preserving noise reduction. This operator is based on the use of a circular mask, the centre of which is known as the nucleus. The brightness of each pixel in the mask is compared to the nucleus. An area of the mask which has similar brightness to the nucleus can then be defined and this area is known as 'USAN' which stands for 'Univalue Segment Assimilating Nucleus'. The local area or USAN contains much information about the structure of the image and is effectively region finding on a small scale. [33] goes on to give mathematical analysis and algorithms for detecting edges from the 'USAN'. In this technique no image derivatives are used and no noise reduction is required. The SUSAN Principle can be stated from [33] as:

'An image processed to give as output inverted USAN area has edges and two di-

mensional features strongly enhanced, with two dimensional features more strongly enhanced than edges.'

This statement gives rise to the acronym SUSAN (Smallest Univalue Segment Assimilating Nucleus).

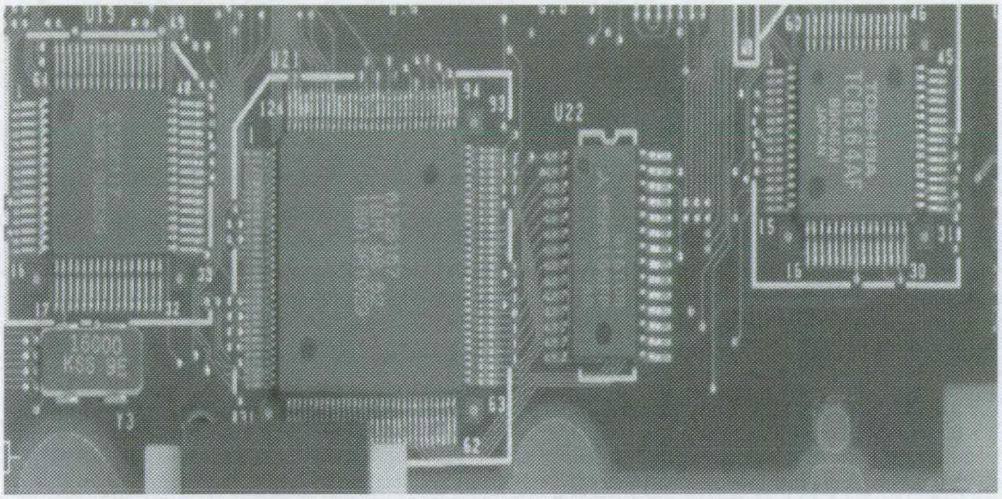

Figure 2.1: *Five IC Grey Scale Image*

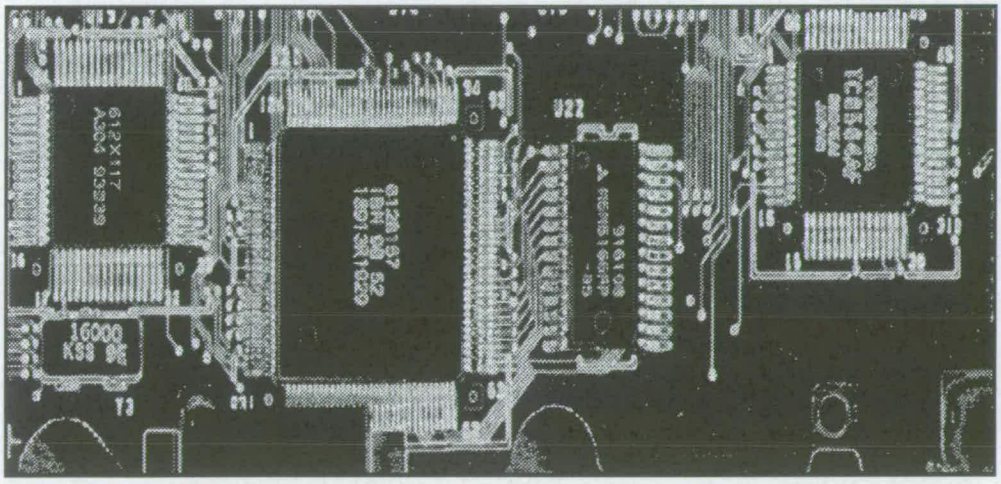

Figure 2.2: *Five IC SUSAN Edge Detection*

An example PCB image is given in Figure 2.1 and the result of the SUSAN operator is given in Figure 2.2. This image shows that the edges of the chips and the chip ID have been detected. These two features are essential to the success of the object recognition and the image registration operators in this work.

2.2.2 The Sobel Operator

The probability processing technique presented in Section 5.5.2 uses the Sobel edge detection operator. This operator is used in this technique because the gradient of the intensity distribution of a test sample is required. The masks for this operator are given in Tables 2.1, 2.2 and 2.3. From these masks the gradient of the intensity in the x and y direction can be computed as given in Equation 2.1 and 2.2. Combining these two results gives the gradient image. A sample result from a PCB image similar to Figure 2.1 is given in Figure 2.3. This image clearly shows that where the greyscale intensity gradient is high, such as on chip legs and chip IDs, there is a corresponding response in the sobel image.

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{2.1}$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{2.2}$$

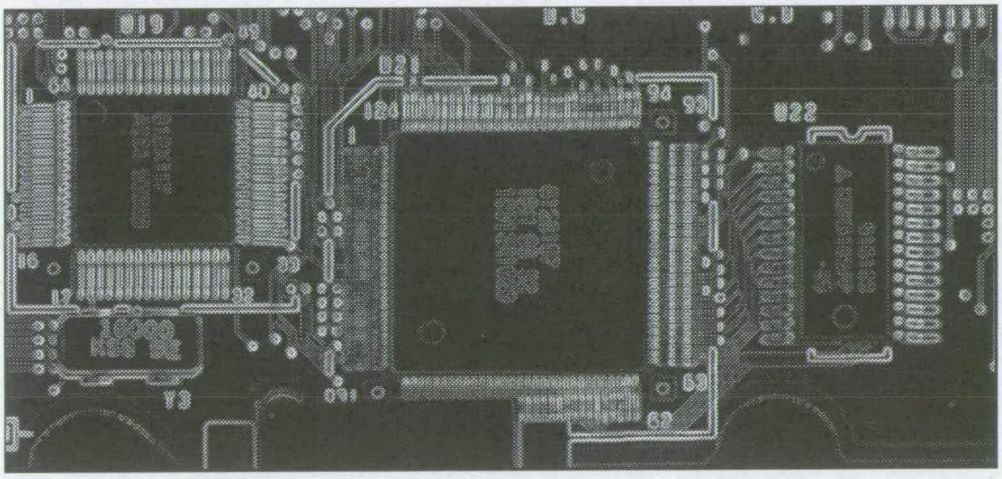

Figure 2.3: *Four IC Sobel Image*

z1	z2	z3
z4	z5	z6
z7	z8	z9

Table 2.1: *3 x 3 Sobel image region*

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

Table 2.2: *Sobel mask to compute Gx*

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

Table 2.3: *Sobel mask to compute Gy*

2.3 Image Segmentation Methods

The main aim of segmentation is to divide an image into parts that have a strong correlation with objects in a scene.

Both [1] and [25] describe a wide range of segmentation techniques. The broad categories are thresholding, edge-based segmentation, region-based segmentation, matching and advanced optimal border and surface detection approaches.

Several segmentation methods have been used in this work which are edge-based segmentation and region-based segmentation techniques. They are Hough Transforms, reduced complexity Hough Transforms and Region Growing. Methods based on Hough Transforms were explored because Hough Transforms can detect shapes where an analytical expression of a shape is known. The borders of integrated circuits correspond to rectangles therefore a Hough Transform approach is suitable. However, the original Hough Transform is complex in terms of both memory requirements and computational load. Reduced complexity transforms were developed in this work but they are not found to give accurate enough results. Region growing techniques are then developed which have low complexity and give accurate object detection.

In edge-based segmentation the edge map from the edge detection stage is processed. This map can be processed in various ways: The map can be processed using a threshold; or using edge relaxation where edge properties are considered in the context of neighbouring edges; or using heuristic graph searching or dynamic programming the border detection process is transformed into a search for the optimal path in a weighted graph - in this final method dynamic programming is an efficient way of simultaneously searching for optimal paths from multiple starting and ending points.

Another edge-base segmentation technique is the Hough transform. The Hough Transform is effective for detecting objects of known shape. This technique can be extended to the generalised Hough Transform, which can be used for detecting shapes where the analytical equations of the shapes are not known.

In region-based segmentation, homogeneity is an important property of regions and is used as the main segmentation criteria in region growing. In region growing the image is divided into zones of maximum homogeneity. The criteria for homogeneity can be based on gray-level, colour, texture, shape or a model (which uses semantic information).

The success of the segmentation stage determines the success of higher level processing algorithms in an image processing system. To achieve effective segmentation and therefore effective object recognition knowledge about the image should be exploited. In an AOI system the CAD data can be used to guide the processing, e.g. when detecting components the CAD data will give their type and co-ordinate positions in the image. From these co-ordinate positions a local search can be performed to detect if the components are present. However, in this work an important class of algorithms have been developed which don't require the use of CAD data to detect components. This is important because in many manufacturing environments accurate CAD data may not be available, particularly when there is a fast turnaround of design, manufacture and test.

The segmentation algorithms in this work are designed to detect the presence of Integrated Circuits (ICs). These algorithms exploit an important feature of ICs, that is they have a Chip ID stamped in their centre. The detection of the Chip ID gives an approximate Chip centroid position which can be used for further processing to find the edges of the ICs.

A contrasting approach to the Segmentation methods used in this work, which do not exploit domain specific knowledge, is given in [35]. This paper tackles image segmentation from

the viewpoint of combinatorial optimisation. A combination of a GA and stochastic annealing algorithm is used to segment images. The performance of the combined techniques is found to be superior than either technique used on its own. The techniques produce accurate results for widely varying images including images of a plane, computer monitor and a phone. However, in this work through exploiting domain specific knowledge significant efficiency gains can be achieved over these techniques.

2.3.1 Hough Transforms

Where objects in an image have known shape and size, segmentation can be viewed as problem of finding the objects within a test sample image. A simple approach to this problem is to move a mask with an appropriate shape and size across the image and then find the correlation between the image and the mask. This solution breaks down where the mask differs to the test sample data due to distortions, zooms, rotation etc. Hough transforms are segmentation techniques which can effectively solve this problem [25].

The original Hough transform is designed to detect straight lines and curves and the method can be used if analytic equations of object borderlines are known. No prior knowledge of region position is necessary. The techniques is robust, where segmentation is not too sensitive to imperfect image data or noise.

The central idea of this technique is to have a transform between image space and parameter space. So for lines represented by the Equation y = kx + q the parameter space would have two dimensions of k and q. Then for each pixel in the image space a voting procedure is performed to points in parameter space. The peaks in the parameter space then give the parameters of potential lines in image space.

The main disadvantage of the original Hough Transform is the memory and computation requirements. For example, for detecting rectangles the parameter space is four dimensional, which correspond to two dimensions for the start position and two dimensions for the width and height. The voting procedure becomes computationally complex as the number of potential rectangles for each point is very large. In Chapter 4 a Hough Transform for detecting ICs is developed, along with a Hough Transform reduced complexity technique. However, these techniques are found to be too complex and memory hungry for the embedded platform used for the project as will be discussed in more detail later.

2.3.2 Region Growing

The segmentation method used in Chapter 5 is based on region growing. The objective of region growing is to partition an image into regions corresponding to an object or part of one [27]. Region growing uses image characteristics to map individual pixels in an input image to sets of pixels called regions. The image characteristic in this thesis works from binary images, with the region growing process forming regions of zero intensity pixels.

Having found the regions in an image, regions are often grouped or merged using higher level domain knowledge or heuristics. In this work the regions to be merged are Chip ID regions. The reason these regions need to be grouped together is to compute approximate chip centroid positions in the absence of CAD data for a PCB. The merging decisions can often be complex, however in this work a simple heuristic is used, that is Chip ID regions are grouped closely together. Through following this heuristic a region growing Chip ID map can be derived, an image which contains Chip ID and virtually no other features.

Images of Printed Circuit Boards are highly complex containing many features other than just Integrated Circuits (resistors, capacitors, tracks, connectors and component IDs for instance) making images of PCBs difficult to segment. The application of region growing in this work does however produce a satisfactory segmentation and the segmentation results can be used to give effective, accurate object recognition.

Region growing is a low complexity segmentation technique suitable for use in a real-time embedded system such as the one described in this work.

2.4 2D Shape Representation Methods

Having performed low level edge detection and then segmentation, shape representation description methods are used to describe the boundaries of the objects in a scene.

The shape representation problem solved in this thesis is 2D. The main 2D shape description methods are: boundary or contour-based shape representation and description; and region-based shape representation and description methods. Contour-based techniques encode the bounding contour of a shape. These techniques include Fourier series expansions of the contour, chain encoding and spline approximations. Region based approaches represent a shape by encoding the 2D space occupied by the shape. Region based techniques include quadtrees,

symmetric axis transform and generalised cones. These techniques are reviewed in [1] and [25].

Important example work in this area is detailed in [36] which present a technique which is both a region and contour based method. In the Smoothed Local Symmetries (SLS) technique primitive curvature discontinuities are defined which form a contour based representation. Primitive region types are also defined. A region is described through a SLS of the primitive region types. Parameters of the region types include a measure of the width of the region and curvature of an associated axis or spines.

The algorithm for computing SLS is complex but the main stages are:-

1) The accurate extraction of the bounding contour and the tangent angle along the contour - using the Canny edge detector for example [34].

2) The measurement of the curvature along curves. This can be computed using k-curvature, Gaussian smoothing, B-spline approximation or least square circle fitting, as detailed in [1] and [25].

3) The finding of 'knots' along the contour by applying the Curvature Primal Sketch (CPS) technique [37]. The CPS is a multi-scale space representation of significant changes in curvature along planar curves.

4) The fitting of circles and lines to sections of the contour between knots.

5) The location of SLS - these are found through testing every contour point against every other to find local symmetries. This final stage computes the spines of a shape and represents the regions of the object.

From the output of the final stage the regions can then be combined to form an object model using a technique such as ACRONYM [36].

In comparison to SLS the shape description of the objects in this thesis is simple. Rectangles are detected in the PCB image which correspond to the borders of ICs. The object representation is therefore a contour based representation consisting of the corner pixel locations of the rectangles.

2.5 Object Recognition Methods

Object recognition takes the results of the 2D shape representation and identifies the objects in a scene. This identification is made through finding a correspondence between part of an image and a particular view of a known object.

There is a large body of research in object recognition methods. A review of the work is given in [38]. Most of the techniques are application specific with knowledge of the task and the environment being implicitly coded into the system. Many of the techniques are for 3D object recognition. The problem solved in this thesis is 2D object recognition.

In [39] and [40] discuss more general issues for object recognition for manufacturing systems.

In [39] the separating of generic knowledge from task specific knowledge in a manufacturing system is studied. This will enable modifiability, extensibility and ease of tailoring for a particular environment. The problems specific to systems which locate parts in a scene include incomplete information supplied by the low level image processing, partial occlusion, and ambiguities which occur with the projection of a 3D scene to a 2D scene. To overcome these problems requires the use of domain knowledge. The use of domain knowledge means the system must be tailored for each part type in the scene. A further problem is the vision system will not be integrated with the databases used by the other components of the manufacturing system. The approach used in [39] to solve these problems is to represent generic objects and reason geometrically about the objects. Further, this generic knowledge is separate from the task specific knowledge which may change. The work describes a framework which addresses the issues of knowledge representation, reasoning and control, and model to image transformations.

In [40] an integrated CAD/CAM and vision system is presented. The recognition of free-form objects from their 2D visual data is described. The work also involves the integration of a CAD system with the object recognition system. The recognition system uses implicit polynomials and algebraic invariants. The data produced by the CAD system has to be converted into a format acceptable by the recognition system. CAD packages use parametric form for representing objects, so a conversion process is necessary to the implicit system used by the recognition system.

In this thesis a reference board is processed to detect the ICs using Region Growing. Through

storing snap shots of the reference ICs a simple block matching algorithm can then be used to perform object recognition on ICs in the test samples.

2.6 Image Registration

The problem of image registration is the process of aligning or overlaying two similar images taken at different times, with different sensor angles or with different sensors. This is a very important process for inspection systems because having aligned or overlayed the two images a difference image can be produced by subtracting the two registered images. This difference image contains information regarding the changes between scenes and is useful in fields such as machine vision, remote sensing and medical imaging [20].

Image registration is also relevant to the application of Automatic Optical Inspection. However, in this work the registration procedure is greatly simplified by using a single fixed camera sensor and an alignment grid for accurately placing test samples. The registration problem is therefore reduced to detecting rotation and displacements of components such as Integrated Circuits. In this work a GA is used for this registration procedure [41] [42].

For a system to be implemented on a production line the registration procedure will have to be carried out to align the captured test sample images. The accurate aligning of the PCBs in relation to the camera sensor carried out for system prototypes may not be possible in a high volume production environment.

The registration procedure can be described using the following Equations 2.3 2.4 [20]. In 2.3 a point (x_1, y_1) is mapped to a point (x_2, y_2). Where s defines a scaling factor, θ defines a rotation angle and t_x and t_y define a translation vector. In this work the rotations are carried out to pixel accuracy. In Equation 2.4 the Equation in 2.3 is rewritten more concisely with p_1 and p_2 the coordinate vectors; t is the translation; s is a scaling factor, and R is a rotation matrix.

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + s \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \tag{2.3}$$

$$\overline{p}_2 = \overline{t} + sR\overline{p}_1 \tag{2.4}$$

In [43] a difference measure of two aligned images is computed. This is not sufficient for the AOI system. In this type of system we want to detect whether the Integrated Circuits are within a tolerance measure not just the difference. This involves a pixel matching procedure. This is carried out with the assumption that the lighting conditions when the images are captured are constant [44]. This is important otherwise the pixel intensities will not match (However, a match could be made within an error band of intensities).

In the image registration procedure a match between a reference and a captured image can be computed by calculating the correlation coefficient given in Equation 2.5 which is with reference to Figure 2.4 [1]. The result of Equation 2.5 can be normalised through dividing by MN.

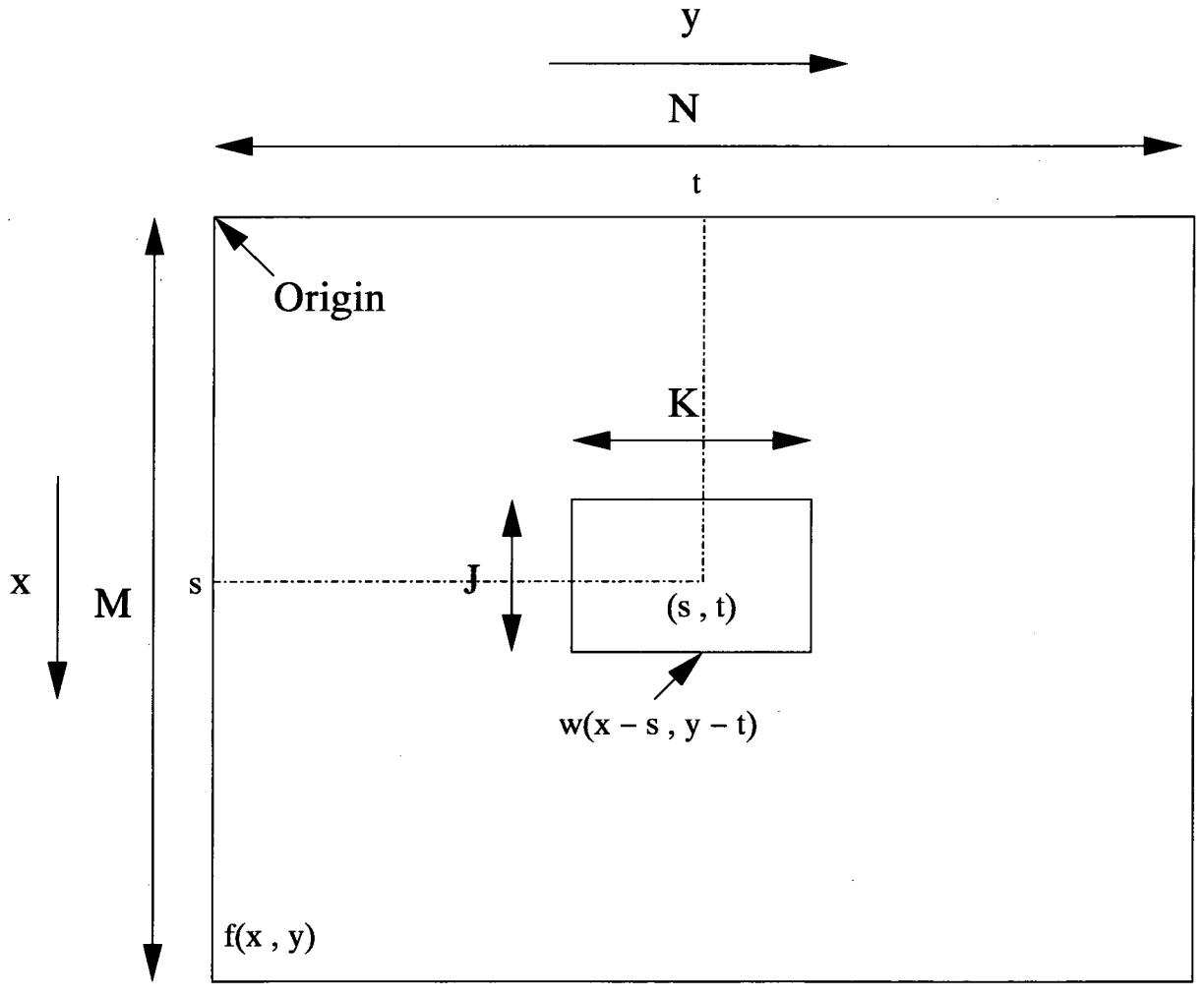

Figure 2.4: *Arrangement for obtaining the correlation of f(x , y) and w(x, y) at point (s, t) [1]*

$$\gamma(s,t) = \frac{\sum_x \sum_y [f(x,y) - \bar{f}(x,y)][w(x-s,y-t) - \bar{w}]}{\left\{\sum_x \sum_y [f(x,y) - \bar{f}(x,y)]^2 \sum_x \sum_y [w(x-s,y-t) - \bar{w}]^2\right\}^{\frac{1}{2}}} \quad (2.5)$$

where s = 0 , 1 , 2 , ... , M - 1, t = 0 , 1 , 2 , ... , N - 1, $\bar{w}$ is the average values of the pixels in w(x , y) (computed only once), $\bar{f}$(x , y) is the average value of f(x ,y) in the region coincident with the current location of w, and the summations are taken over the coordinates common to both f and w. The correlation coefficient γ(s, t) is scaled in the range -1 to 1, independent of scale changes in the amplitude of f(x, y) and w(x ,y). The higher value of intensity of γ(s , t) is in the position where the best match between f(x , y) and w(x , y) is found.

The problem with this method is it becomes computationally expensive for arbitrary rotations and displacements. Looking for the best match between reference and captured image involves exhaustive rotations of w(x, y). In the AOI problem detection of arbitrary rotations and displacements of Integrated Circuits has to be solved. Therefore in this work a new method of detecting rotations and displacements has been developed which uses a search based on a GA [42] [41]. This method is found to be robust and efficient fór highly complex test samples.

2.7 Evolutionary Algorithms and Genetic Algorithms

The term evolutionary algorithms covers a range of techniques including GAs, evolutionary strategies and evolutionary programming [31] [32] [28] [29] [30]. They have one fundamental commonality: they involve reproduction, random variation, competition, and selection of contending individuals in a population.

GAs were first proposed in [30]. There are three features which distinguish this type of algorithm from other evolutionary algorithms. Firstly, the representation of solutions is usually binary strings. Secondly the method of selection for survival of a string is proportional selection. Finally, the primary method of producing variations is crossover. This final feature makes GAs distinctive. Many authors have used alternative methods of selection and many have used other variations on binary strings for representation. There have also been many alternatives to crossover proposed, but they all have as a basis the analysis of schemata and building blocks as presented in [30]. Before schemata and building block processing can be explained some terms need to be defined along with the basic structure of a GA.

The individual structures in a GA's population are referred to as chromosomes and are typically binary strings. These are the genotype structures which are manipulated by the algorithm. The fitness evaluation decodes the structures into a phenotype and assigns a fitness value. The value at each locus on the chromosome is referred to as an allele. Individual loci may also be called genes. Genes may also be combinations of alleles that have some phenotypical meaning.

2.7.1 The Basic Genetic Algorithm and its operators

The basic structure of a GA is given in Pseudo code 1. The initial population of binary coded strings is usually set to be random giving a distribution across the search space. Each initial solution genotype is then evaluated according to a fitness measure giving a phenotype. In this work the genotype structure encodes a rotation in degrees and x and y direction offsets in pixels for images of Integrated Circuits. These rotations and displacements are then applied to a captured image to match them to a reference image. It should be noted that rotations and displacements are carried out at pixel resolution. The fitness is the number of pixels that match divided by the total number of pixels in the sub-image of the Integrated Circuit.

Begin
Generate an initial population of binary coded solutions
Derive a fitness measure for each solution
Repeat until convergence
{
Repeat to create new population
{
Select two parents according to their fitness
Mate parents in order to create two offspring (Crossover)
Mutation
Use offspring to create a new population
}
Derive a fitness measure for each solution
}
End

Pseudo code 1: *Genetic Algorithm*

The main loop of the algorithm then begins with two parents being selected according to their fitness for reproduction. There are many types of selection methods [28] and in this work roulette wheel selection is used. In this method a biased roulette wheel is created where each

string in the population has a roulette wheel slot sized in proportion to its fitness. To select parents the roulette wheel is spun and reproduction candidates are selected according to their fitness. Using this approach highly fit strings are selected more often than low fitness strings.

The strings selected are then mated using a technique known as crossover. The aim of the crossover operator is to combine partial solutions to give fitter strings. The type of crossover used in this work is known as single point crossover. The crossover point is selected randomly along the length of the string and then portions of the two strings are swapped. This is best explained by an example Figure 2.5

Not all pairs of strings are selected for crossover. The crossover rate will be selected by the user at the start of the algorithm run. Typical rate of crossover is 60 % of strings per generation and this rate is used in this work [32].

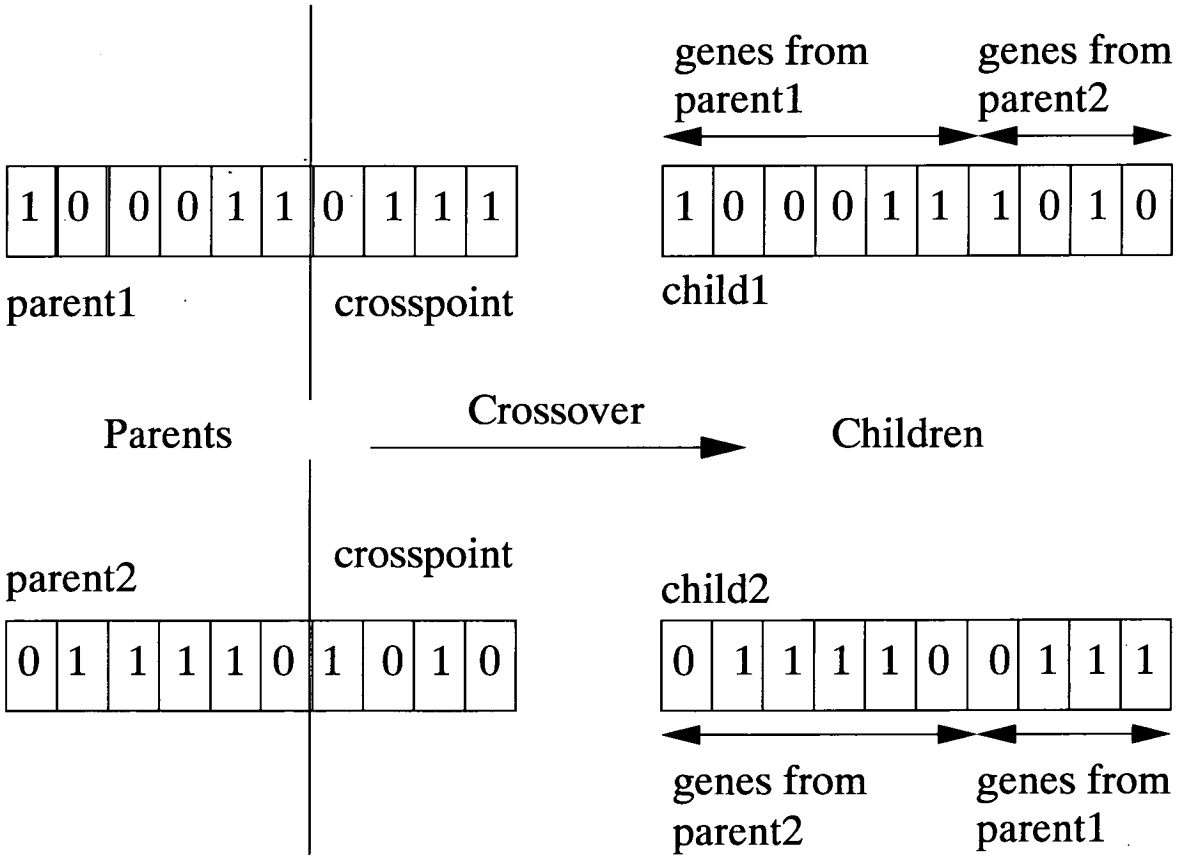

Figure 2.5: *Example Crossover*

A mutation operator is then applied. This operator usually involves the flipping of a single

bit in a chromosome. The process of reproduction and crossover may cause the loss of some potentially useful genetic material. Mutation is effectively a random walk through string space and ensures against this loss. The mutation rate is set by the user at the start of the algorithm run and is application specific (it depends on the size of the population and the length of the chromosomes). A typical mutation rate would be to modify one bit per thousand bits as defined in [32]. The ideal rate for the application in this work is found through experimentation to be higher, set at 5 per cent of bits per population. This higher rate of mutation is used otherwise the computation tends to get stuck in local minima with the whole population of chromosomes giving the same fitness value. The population size used is 50 strings and the chromosome length is 66 bits. This gives a total population size of 3300 bits. With a 5 % mutation rate 165 bits are mutated per generation. Specific experiments giving the time for the GA to converge are given in section 6.5.

The algorithm now progresses with the two strings inserted into the new population and this loop continues until a new population has been created. The fitness measure is then applied and the whole process is repeated until convergence criteria has been met. In this work the convergence criteria is the matching of all the captured integrated circuit images to the reference images. This criteria shows that the ICs have been registered and are within tolerance with respect to rotation and displacement.

In this work the basic GA described here is enhanced with a hill climbing operator and the elitism operator. Through the use of a hill climbing operator faster convergence of the algorithm is achieved. The types of hill climbing operator discussed in [31] are Steepest-ascent hill climbing (SAHC), Next-ascent hill climbing (NAHC) and Random-mutation hill climbing (RMHC). The method used in this work differs from these algorithms. Firstly, a candidate solution is not chosen at random, hill climbing is performed on up to 20 per cent of population members (This is an upper bound: hill climbing is not guaranteed to find a string with higher fitness value). Secondly, each bit is not mutated in the string. Instead, the string is first decoded into a rotation and a X-Y displacement. This phenotype is then modified by an increment or decrement of a single degree rotation or a single pixel displacement in the X and Y directions. At each stage the fitness function is then recomputed. If a higher fitness is achieved then this new phenotype is re-encoded and inserted back into the population replacing the original individual on which the hill climbing is performed. The hill climbing operator can therefore be thought of as an extra search operator which reduces the amount of time the algorithm takes to converge.

The elitism operator ensures that the fittest member of the population survives to successive generations. In the work described here the top 10% of individuals are guaranteed to survive to successive generations. They replace the individuals with the lowest fitness in the population. The use of elitism reduces the amount of time the algorithm takes to converge.

The major features of GAs can be summarised as follows:-

1) GAs work with a coding of the parameter set, not the parameters themselves.

2) GAs search from a population of points, not a single point.

3) GAs use an objective or fitness function, not derivatives or other auxiliary knowledge.

4) GAs use probabilistic transition rules, not deterministic rules.

2.7.2 Schema Theory

Having given a brief description of a basic GA and its use in this work the very important schema theory can be described which is the fundamental theorem of GAs [32]. This analysis is directly relevant to the work in Chapter 6 because a GA is described which is based on the Simple Genetic Algorithm (SGA) described in [32]. SGA uses a proportional selection method for reproduction (roulette wheel selection), binary strings as chromosomes, single point crossover, and single bit mutation.

A notation is used to define schema as follows: schema H taken from the three-letter alphabet V+ = 0 , 1 *. The * symbol is a wild card symbol which matches either a 0 or a 1 at a particular position. For example, the bit string schema H of length 8 11**01**. The string 11000111 is an example of this schema H, because the string alleles a_i match schema positions h_i at the fixed positions 1,2 and 5,6.

Schema have two important properties: schema order and defining length. The schema order denoted by o(H) is the number of fixed positions present in the template. Therefore, for the previous example the order is 4. The defining length is the distance between the first and last string position, denoted by δ(H). In our example string the defining length is 5.

The effect of reproduction, crossover and mutation on schemata within a population of strings can now be analysed.

The reproductive schema growth equation can be written as Equation 2.6:

$$m(H, t+1) = m(H, t)\frac{f(H)}{\bar{f}} \tag{2.6}$$

This equation shows that a particular schema grows as the ratio of the average fitness of the schema to average fitness of the population. Schemata with fitness values above the population average will receive an increasing number of samples in the next generation and schemata with fitness values below the population average will receive a decreasing number of samples. Further, [32] goes on to show that reproduction allocates exponentially increasing numbers of trails to above average schemata and conversely exponentially decreasing numbers of samples to below average schemata. However, reproduction alone does not promote exploration of new regions of the search space, this is the point of crossover. Crossover is a structured yet randomised information exchange between strings. Crossover creates new structures with a minimum of disruption to the allocation strategy dictated by reproduction alone. This results in exponentially increasing (or decreasing) proportions of schemata in a population. [32] shows the effect of crossover on strings defined using the three letter alphabet V+. The probability of survival under simple crossover is given by the Equation 2.7.

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{l-1} \tag{2.7}$$

In this equation p_s is the probability of survival of a schemata, p_c is the probability of crossover, and l is the length of the string. The equation shows that the probability of the survival of schemata depends on the defining length of the schemata. The schemata will be disrupted whenever a cross site within the defining length is selected from the l - 1 possible sites. This shows that short defining length schemata have a higher chance of surviving crossover.

In order for the schema H to survive mutation all the specified positions must survive. A single allele survives with probability (1 - p_m). A particular schema survives when each of the o(H) fixed positions within the schema survives. The schema survival probability may therefore be approximated by the expression 1 - o(H) · p_m for small values of p_m.

The final equation defining the effect of reproduction, crossover and mutation is Equation 2.8

$$m(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{\bar{f}} \left\{ 1 - p_c \frac{\delta(H)}{l-1} - o(H) p_m \right\} \tag{2.8}$$

Here Equations 2.6 2.7 have been combined along with the expression for the effect of mutation.

Given this analysis and Equation 2.8 the schema theorem or the fundamental theorem of GAs can be stated as follows: highly fit, short-defining-length schemata or building blocks are propagated generation to generation by giving exponentially increasing samples to the observed best. Although this does not mean that GAs are guaranteed to always converge to a global optimum, it has been found through experimental studies that they will usually converge to a near global optimum for problems with large complex search spaces (including image registration problems, such as the one defined in this work and [43] [45] [46]).

2.8 Genetic Algorithms for Image Registration

Having described the computation carried out by a GA, it is now possible to describe how the technique has been applied to the problem of image registration. The original work in this area was by [47] with an application to medical imaging. Further work in the area of medical imaging is given in [43] and [45]. Further application areas are detailed in [43] and [48] for remote sensing, and [49] for finger print analysis. The results from these works show that the application of a GA to image registration is robust, accurate and fast and is applicable to a variety of problem areas.

The work by [47] states that a search procedure for image registration is required to find an optimal point in multi-dimensional parameter space. This optimal point corresponds to a set of parameters which minimises a distance function between a reference and captured image. The search is hindered due to the size of the search space and the presence of many local minima. The use of a GA is found to give a good approximate solution with reasonable computational effort. This result is achieved even though the optimisation problem addressed is complex. This is due to the images being registered containing elastic motion as well as rotation and translation, which gives a more complex search space.

GAs are task independent, for the work in [47] the main task dependency is the fitness function. The fitness function gives a numerical rating for each point in the search space. The paper

presents three methods of computing the fitness function: the sum of the absolute values of pixel differences; the sum of the squares of the pixel differences; summing average pixel differences over recursively smaller sub-images. This final method was found to be the most effective. In the final method the GA concentrates initially on finding transformations which provide a good match with respect to global features, while focusing later on finer detail.

The computation time used by a GA to register images is an important issue. In [50] a theoretical study is presented. The study explores the relationship between the amount of effort spent on individual evaluations and the number of evaluations performed by the GA, for an image registration problem. The major conclusion from this work is the overall efficiency of the GA may be improved through reducing the time spent on individual evaluations and increasing the number of generations performed. The amount of time spent on each evaluation can be reduced by sampling the images rather than computing a difference measure for every pixel.

In [45] the problem of registering two 3D medical images of a heart, vertebrae or fetus, is addressed. The chromosome structure is similar to that presented in this thesis, except a 3D as opposed to a 2D translation and rotation is coded. The fitness measure is the Euclidean distance [51] between each correspondence pair. The use of a Genetic Algorithm to search the space of transformations is found to be robust and faster than other methods. These other methods rely on there being a good initial guess of a match, or require the user to give prior information on correspondence or specific feature points. This paper also samples the images to reduce computation time of the fitness function.

The work presented in [52] is an important basis to the work presented in this thesis. This work present the use of a customised VLSI hardware platform for executing a Parallel Genetic Algorithm (PGA).

PGAs can be divided into three types, 'Standard' , 'Coarse grained' and 'Fine grained'. In a 'Standard' parallel GA the processing of a population is distributed over a number of processors. With a 'Coarse' grained GA several populations of genes are run in parallel. After a number of generations the separate populations export a set of individuals to neighbouring populations. 'Fine' grained GAs act on each member of the population in parallel. Each member of the population performs crossover with its immediate neighbours. The neighbourhood is defined by a topology and a distance parameter.

The work [52] gives results for a 'Fine' grain GA with massive parallelism on a single VLSI

chip. Results show that images are registered within 35 generations of the GA, recognising an image in a tenth of a second. This result shows the potential of a GA based Image Registration system when implemented on customised VLSI hardware.

2.9 Genetic Algorithms for Real-time Embedded Systems

For a hard real time system, where deadlines for computation have to be met, GAs may not be suitable because of their variable execution time to convergence. However for a soft real time system, such as the one described in this work, they are found to be robust and efficient.

The main disadvantage of the GA used in this work are that for every member of the algorithm's population an evaluation has to be made, which consists of matching a sub-image of a reference to a sub-image of a captured image. However, the algorithms presented in this work can be made computationally less expensive by reducing the number of pixel transformations made in the GA's fitness function computation. This is possible because the corners of the Integrated Circuits can be found by using object recognition techniques proposed in Chapter 5. When computing the fitness function transformations need only be performed at the corners of the ICs.

2.10 Evolvable Hardware

2.10.1 Gate level and Function level Evolvable Hardware

The basic idea of EHW is the hardware structure adapts itself to the environment in which it is embedded. To achieve this a GA is used to derive chromosomes for configuring Programmable Logic Devices (PLDs) and Field Programmable Gate Arrays (FPGAs) embedded in a system. An overview of this work on adaptive embedded systems is given in [19] and [53].

EHW is designed to adapt to changes in task requirements or changes in the environment through reconfiguring its own hardware structure dynamically and autonomously. An important feature of these systems is the hardware is adapted in real-time.

Existing work on EHW concentrates on adaptation at the circuit level. There are two main approaches to the design of circuits, at the low level AND and OR gate level [19] or at the functional level [54] [55] [56].

The closest gate level application presented in [19] to the work presented here is the data compression chip for Electrophotographic Printing (EP) [57] [58] [59]. In this application a GA is used to search for a set of optimal templates which are used to reconfigure a hardware prediction mechanism. The data compression chip consists of two parts, a RISC processor and the data compressor. The RISC processor controls the data compressor, runs the GA calculations, and interfaces with the host computer. The data compressor receives the optimal template identified by the GA, compresses the input image and returns the size of compressed data to the RISC chip for GA evaluation. This compression chip satisfies the requirements for the processing speed of EP printers.

The size of circuits that can be designed at the AND and OR gate level as the GA runs is limited. This restricts the usefulness of circuits for practical applications. Larger and more useful hardware functions can be evolved through evolving circuits from high-level hardware functions such as adders, subtracters and sine generators etc.

An application with a function level evolvable hardware device is presented in [54]. In this system a RISC processor is combined with a set of 15 configurable DSP processors. This architecture is also suitable for an embedded system for real-time applications. However, this approach was not followed in this work due to the cost of the hardware platform.

2.10.2 Use of Microprocessors for an Evolvable Hardware System

The EHW system developed in this work differs from existing literature on EHW. In the AOI application standard programmable cores are used with an adaptive software system implemented in software. The reasons for using standard programmable cores as opposed to FPGAs or custom logic are given in [60] and they can be summarised as follows: In digital design using a predesigned instruction set processor may result in more efficient execution of the software than designing custom logic. There are two main reasons for this, firstly microprocessors execute programs very efficiently. Modern RISC processors execute one instruction per cycle. There is an overhead for interpreting instructions but this can be hidden by using pipelining techniques [61]. Secondly, microprocessor manufacturers invest heavily in making their CPUs run fast. The resources spent on developing a new microprocessor are much greater than resources available to develop custom logic. For microprocessors the latest VLSI technology is used and if the custom logic is designed using slower circuits it will mean its performance advantage is negligible.

For the AOI application presented in this work highly complex algorithms are run on standard programmable cores. Through executing them on microprocessors considerable flexibility is gained, whereas if custom logic is designed, it cannot be used for other functions. In this work a prototyping methodology was followed and the use of microprocessors gave great flexibility because software updates could be developed and downloaded and run with a fast code, test, debug cycle.

This approach is also examined in the work by [62] where Application Specific Instruction set Processors (ASIPs) are favoured over Application Specific Integrated Circuits (ASICs). The principle advantages of this approach are multiple related applications as well as different generations of an application can be mapped onto the same ASIP. The ASIP has a shorter time-to-market as writing software is less expensive than designing a hardware solution. However, general purpose programmable solutions as used in the AOI work are unacceptable for many applications as they have a power/delay overhead. For ASIPs the solution is to use application specific customisation through specialised hardware resources. This places significant requirements on the software development environment for the processor development as well as for the application code. In [62] a set of tools to meet these requirements are described.

2.11 Existing AOI Methods for PCB Analysis

There is a significant amount of work published in papers and patents on AOI systems for PCB analysis.

2.11.1 Published Work on AOI

AOI systems for PCB analysis are used to detect solder joint problems as well as component placement inaccuracies. The work presented in [63] uses novel structured-lighting inspection technology. A camera catches the reflection of the solder joint surface caused by the lighting. A slant map surface shape estimation technique is then developed for the solder joint. The map extracts the shape information of a solder joint, which is based on the slant angle of the solder joint surface. From the slant map joints can be classified into categories, including good joint, bridged solder joint, lacking solder, surplus solder and open joint.

The work of [64] gives an alternative approach to image registration for detecting alignment

faults of surface mount technology devices. The technique detects lead displacement with respect to the components ideal position. Information from many leads is fused into stochastic framework for accurate displacement estimation. The approach exploits the fact that individual leads encode the same information regarding relative positioning of the rigid body of the component on the pad area. Complementary information from all leads is fused into a Bayesian estimation framework [65].

In [66] a complete AOI system is presented which uses referential methods. In referential methods a test PCB image is subtracted from a reference image to generate a difference image. This technique is computationally efficient but relies on very accurate alignment of test samples. This work presents a method for aligning images accurately. This method uses a full search block matching algorithm. The positions of blocks in the reference image which contain a distinct feature pattern are selected. The algorithm then computes the sum of the absolute differences between the blocks in the reference and test image. The algorithm iteratively computes and minimises the sum of the absolute differences for a small range of rotations. The rotation which minimises this calculation is the best estimate of rotational displacement between the reference and captured image. This result can be used to align the images before computing the differences.

In [66] the PCB images are thresholded so objects of interest are separated from the background before computing the difference measure. The problem with this method is computing the optimal threshold. This is a problem because of the variation in illumination across a test sample. A shading correction algorithm is presented which computes an optimal threshold. This algorithm uses regions from the unpopulated 'clean copper' image and the PCB reference image. The 'clean copper' image is transformed into a shade-corrected image according to a linear gray level transformation. Each pixel value of the shade-corrected image is used as a threshold for the binarization of the corresponding pixel of the reference image. This technique produces effective binarization of the reference image.

2.11.2 Patents on AOI

A sample patent for a complete AOI system for diagnostics of PCBs is given in [67]. In this system each scanned pixel is compared against a corresponding database and given a defect weight in accordance with preset values for a given pixel state. Tolerances are defined for each individual PCB feature. The tolerance database has at least three states in which each colour

is weighted and adjacent pixels are grouped into arrays or "bins". An error signal is generated when the sum of the pixel weights in a bin exceeds a preselected threshold.

Another complete AOI system for PCB analysis is detailed in [68]. This system carries out dimensional verification of components and pattern recognition using template matching.

Other patents which detail address different aspects of AOI systems are [69] and [70]. In [69] the problem of efficiently entering the CAD data which describes a reference board is addressed. In [70] a specialist test jig for holding PCB test samples and a camera is described.

The patents described in [71], [72] and [73] address the problem of camera set-up and lighting in an AOI system. Effective image capture and sufficient illumination is critical to an accurate diagnostic process.

2.12 Summary

In this Chapter important machine vision and evolutionary algorithms relevant to this work have been discussed. This Chapter has shown that significant efficiency gains can be achieved for an inspection system through the use of domain specific knowledge and low complexity segmentation techniques such as Region Growing.

In this Chapter image segmentation methods for object recognition have been reviewed. This review includes two edge detection methods used in this work, the Sobel and SUSAN operators. Edge detection is essential before segmentation methods such as the Hough Transform and Region Growing can be applied.

The object recognition task solved in this thesis is a 2D problem of detecting ICs in a PCB image. This Chapter goes on to review existing work in 2D shape representation and object recognition.

In this Chapter as well as machine vision algorithms, evolutionary techniques are discussed, focusing on GAs. GAs are used as the basis for a genetic search for the image registration system presented in Chapter 6. The basis of the success of the operation of GAs is schema theory. Important equations for schema theory relating to crossover and mutation genetic operators are given which leads to the fundamental theorem of GAs. Methods of improving the efficiency of GAs through operators such as elitism and hill climbing are detailed.

In this Chapter existing work on the application of GAs to image registration problems is detailed. The use of a GA was found to be robust, fast and accurate for a variety of applications including medical imaging, finger print analysis and remote sensing. Further work detailing the results of using a PGA running on a customised VLSI chip, show the potential for the techniques presented in this thesis.

In this Chapter approaches to EHW, and how the EHW system developed for this work differs from the main body of literature on EHW, is discussed. The section gives important reasons why standard programmable cores (SOC and DSP) have been selected for this work over using customised logic, a FPGA or PLD or ASIC for instance.

This Chapter concludes with a description of existing work published in the area of AOI systems for PCB analysis. Patents filed on AOI systems are also detailed.

Chapter 3
Real-time Systems and System-On-Chip technology

3.1 Introduction

This chapter introduces issues in real-time and embedded systems relevant to the development of AOI systems. The Chapter defines real-time and embedded systems and how they differ from many workstation based applications. This Chapter also details the extra demands on the development process when compared to developing application software for a workstation.

The important area of Hardware/Software Co-design is introduced where components of a design are selected to run in either hardware or software to give optimal performance. As technology has progressed it is now possible to integrate the components of a system onto a single chip, known as an SOC. A definition of SOC is given and a system-level design methodology for developing systems for SOCs is then described. This description is with direct reference to the AOI system development. The Chapter concludes with a description of existing work in vision systems targetted for SOC devices.

This chapter is structured as follows: in Section 3.2 issues in real-time and embedded systems are discussed; in Section 3.3 discusses Hardware/Software Co-design; in Section 3.4 a definition of SOC technology is given; in Section 3.5 issues relating specifically to the development of embedded software for SOC are discussed; in Section 3.6 methodologies for SOC development are discussed; in Section 3.7 existing literature on the use of SOC technology for vision systems is presented; and Section 3.8 summarises the chapter.

3.2 Real-time Systems

Real-time systems are computer systems where the meeting of deadlines for computations is an overriding factor. These systems monitor, respond to, or control an external environment

[7–11]. The environment is connected to the computer system through sensors, actuators, and other input-output devices.

3.2.1 Hard and Soft Real-time Systems

Real-time systems have deadlines where a response to an event in the environment of the system has to be made in a fixed amount of time. In a hard real-time system if this deadline is missed it may cause a catastrophic failure in the system's environment. Examples of this type of system are Air Traffic Control Systems [8] or a control system for a nuclear power plant [74]. The second type of real-time system can be classed as a soft real-time system where the missing of a deadline causes degradation of system performance rather than failure. The AOI system presented in this work is of this type of real-time system.

3.2.2 Embedded Systems

Embedded systems are components of a larger system. A further feature of Real-time systems is they often run on embedded hardware. The embedded hardware for Systems of this type is usually customised and has area and power consumption constraints. The engineering task for an embedded system is therefore greater than development for a workstation or PC, as often memory and processor resource are restricted. The system developed in this work is an embedded system.

3.2.3 Real-time Systems and Concurrency

Real-time systems are often concurrent. Real-time systems must deal with the inherent physical concurrency that is part of the external world to which they are connected. However, for the prototype AOI development the code is sequential. This simplified model of computation is adequate for the type of processing the system performs, with a reference board being processed and then processing of a batch of test samples sequentially.

For the AOI system the system is partitioned between a micro-controller and a DSP. For this computation the genetic algorithm runs on the micro-controller while the computationally expensive parts of the genetic algorithm run on a DSP, the fitness function for example. Communication between the two processes occurs using message passing via the host PC. The

execution is still sequential however, with the micro-controller sending parameters to the DSP and then waiting for the DSP to pass back results of the fitness function before continuing execution.

Real-time embedded systems often require the integration of a Real-Time Operating System (RTOS) [8, 10] into the system. RTOS are similar to those for a workstation in that they manage hardware and software resources and provide services to users. However, for real-time systems there is an extra requirement which is predictability with respect to time. All services must be executed within bounded and known times, and at times that are controlled and known. However, this is not a critical requirement for the AOI system because the system is a soft real-time system. Therefore, for the prototype systems for this work it was not deemed necessary to integrate a RTOS into the system.

For the final prototype of the AOI system, integrating a RTOS will be necessary when the hardware platform migrates to a full SOC implementation. The RTOS will handle the distribution of the computational load across the programmable cores in the SOC. The operating system will also handle inter-process communication and process priority. The system will be further complicated by the need to have a process monitoring the input from the user.

3.3 Software/Hardware Co-design for Embedded Systems

A key feature of system development for embedded systems is Hardware/Software co-design. Components are implemented in Software for flexibility. Components are implemented in Hardware for speed. Optimal trade-off between hardware and software implementations can be found through simulation using system-level design tools and methodologies such as VCC and HASoC [2, 75]. Optimal trade-off varies from application to application. Further related issues with many embedded systems is power consumption and area restrictions.

An example of successful hardware software co-design with benefits in terms of processing speed and power consumption for a MPEG-2 video coding/decoding application are given in [76]. MPEG-2 coding/decoding algorithms are used for multiple different applications such as set top boxes and digital cameras. Through combined hardware and software implementation 40% higher decoding speed and 36% lower power consumption is achieved. These benefits are found when compared with an entirely software based implementation or a hardware based solution using ASIC custom chips. The partitioned system is described in high-level Veri-

log/VHDL [77] hardware description language for implementation in a Virtex 1600E Field Programmable Gate Array (FPGA). The system comprises a 32-bit RISC processor [61] and dedicated circuits for performing functions such as the Discrete Cosine Transform (DCT) [76].

As technology has progressed it is now possible to integrate all the components of a complete product onto a single chip, which is known as an SOC. This has led to system level design methodologies for Hardware/Software co-design for SOCs.

3.4 System-On-Chip: a definition

In this work an SOC architecture is used as the basis for a high performance Evolvable Hardware platform for executing complex image processing functions. The standard definition of SOC is as follows:

> 'System-On-Chip is defined as a complex IC that integrates the major functional elements of a complete end-product into a single chip or chip-set. A SOC design incorporates a programmable processor, on chip memory, and accelerating function units implemented in hardware. It also interfaces to peripheral devices and/or the real world. SOC designs encompass both hardware and software components. Because SOC designs can interface to the real world, they often incorporate analogue components, and in the future also include opto/microelectronic mechanical system (O/MEMS) components' [18].

3.5 System-On-Chip Technology and Embedded Software

3.5.1 System-On-Chip Technology and Software Development Demands

The development of SOC technology requires new approaches to embedded software development [18, 78]. This new approaches are required due to the fast time-to-market pressures of the consumer market. SOC level of integration requires the development of a stable hardware kernel in an SOC platform, with a wide number of product derivatives through software customisation. Further demands on the process are due to limitations on size, cost and power consumption of the device.

3.5.2 System-On-Chip and Software Re-use

Given these constraints the main feature of embedded software development for SOC devices is the 're-use' factor. This issue is critical to achieve fast time-to-market. In traditional embedded software development the use of totally hardware dependent, optimised C and assembler code is common. Code is engineered in this way because of stringent memory and performance goals. Software of this form is only re-usable under strictly limited circumstances i.e. product derivatives using the same processor and algorithm with the same memory, performance goals and identical interfaces to drivers and the memory-map.

The re-use driven approach to embedded software takes a totally different perspective on software development. This approach relies on the following: clean packaging of software components with well defined external interfaces; isolation of all direct contact between the software and the hardware machine in a clean hardware abstraction layer; use of Application Programming Interfaces (APIs) to the machine dependent software layers to remove dependence on hardware specific features; use of object oriented languages such as C++ [23]; modelling tools such as VCC [75] to support partitioning of functions between hardware and software to achieve system optimisation; use of a high level modelling notation such as UML for real-time [9, 79].

It should be noted that this new approach to re-use may provide less optimality in a specific implementation. However, across repeated re-use in derivative products much faster time to market will be achieved.

3.5.3 Software Re-use and AOI Systems

When developing for AOI systems an approach supporting re-use was followed with platform independent image processing code developed in C and C++. This code used existing code libraries such as an image processing library in C++ [21] and a genetic algorithm implementation in C [32]. Cross platform compilation was carried out with the code being targetted from a Sun UNIX host to embedded processors including an ARM7 SOC [80] and a Texas EVM6201 advanced Digital Signal Processor (DSP) [81–83] . This compilation was made with very few changes to the code and shows that a conversion to alternative processors and therefore re-use could be achieved in little time. It should be noted that the most significant code change for the ARM implementation was the modification of the image rotate code used in the image regis-

tration procedure from a floating point to an integer implementation, this optimised execution time (this change is detailed in full in section 6.6).

3.6 System-On-Chip Methodology

Heterogeneous systems which comprise a mix of programmable and dedicated components designers rely on methodologies and tools that allow them to explore their designs at the system-level [84]. Designers build models of the embedded system to verify constraints such as computing time and power consumption. The more accurate a model is the greater the time needed for model development and simulation. In order to reduce the time needed for modelling and simulation, evaluation of design choices should be at the early phases of the design process. At more detailed levels of abstraction the opportunities to explore alternative designs are reduced. Therefore, methodologies to deal with exploration of design choices for embedded systems at the system level are very important.

Methodologies are important for this work because the software components developed will become part of larger system as the work progresses from research to development. This development will involve the design of a SOC with the software components optimised for the new target. Being able to explore the design of this integrated platform at the system level with reference to the performance targets of the system will be essential.

3.6.1 Hardware Software Objects on Chip: the HASoC Methodology

The work in [2] presents Hardware Software Objects on Chip (HASoC) which is a new methodology which uses an object-oriented approach to the development of complete embedded systems (including hardware and software platform, and application). This methodology is a good example of a contemporary SOC methodology which allows developers to explore designs at the system level. HASoC raises a lot of issues relevant to the development of the AOI system described in this thesis. This methodology will therefore be described in some detail with reference to the implications for AOI development where relevant.

The authors of [2] build on their existing methodology which is the MOOSE method. The MOOSE method starts with an abstract behavioural model of a system in which components are not committed to an implementation in hardware or software. System functionality is validated

through execution of the behavioral model. The model is then partitioned into appropriate technologies (the committed model). The aim of this partitioning is to meet the products design constraints and optimise its design.

MOOSE provides strong support for the automatic generation of a top-level description of a platform. Executable modelling is very valuable for understanding a system's behaviour, and enabling the system functionality to be verified earlier in the development cycle. This capability significantly reduces the detailed implementation work required in the later stages of the lifecycle. This work is important because design deficiencies are captured earlier in the design cycle, lowering the likely maintenance cost of the system [7].

In [2] the authors highlight a number of important deficiencies in the MOOSE method. These deficiencies highlight common problems with this type of methodology and restrict their usefulness in a wide variety of applications. The authors go on to present the HASOC method which addresses these issues. The method uses UML notation with minor extensions. The method introduces an iterative, incremental lifecycle. The main stages in the lifecycle are given in Figure 3.1.

3.6.2 The HASoC Methodology: key stages

3.6.2.1 Product Concept

This stage defines the product in an informal manner. The aim is to specify the functional and non-functional requirements and the system platform. The non-functional requirements can include performance, power consumption and area constraints. The non-functional constraints of an AOI system include the time it takes to process a PCB and the static, executable code size for instance, and dynamic memory requirements of the software (which includes the storing of images of PCBs). For the AOI application issues such as power consumption and SOC component size are not as critical as for a portable or battery powered system, such as a mobile phone. The product concept specification will be made in natural language and doesn't have to be complete at this stage.

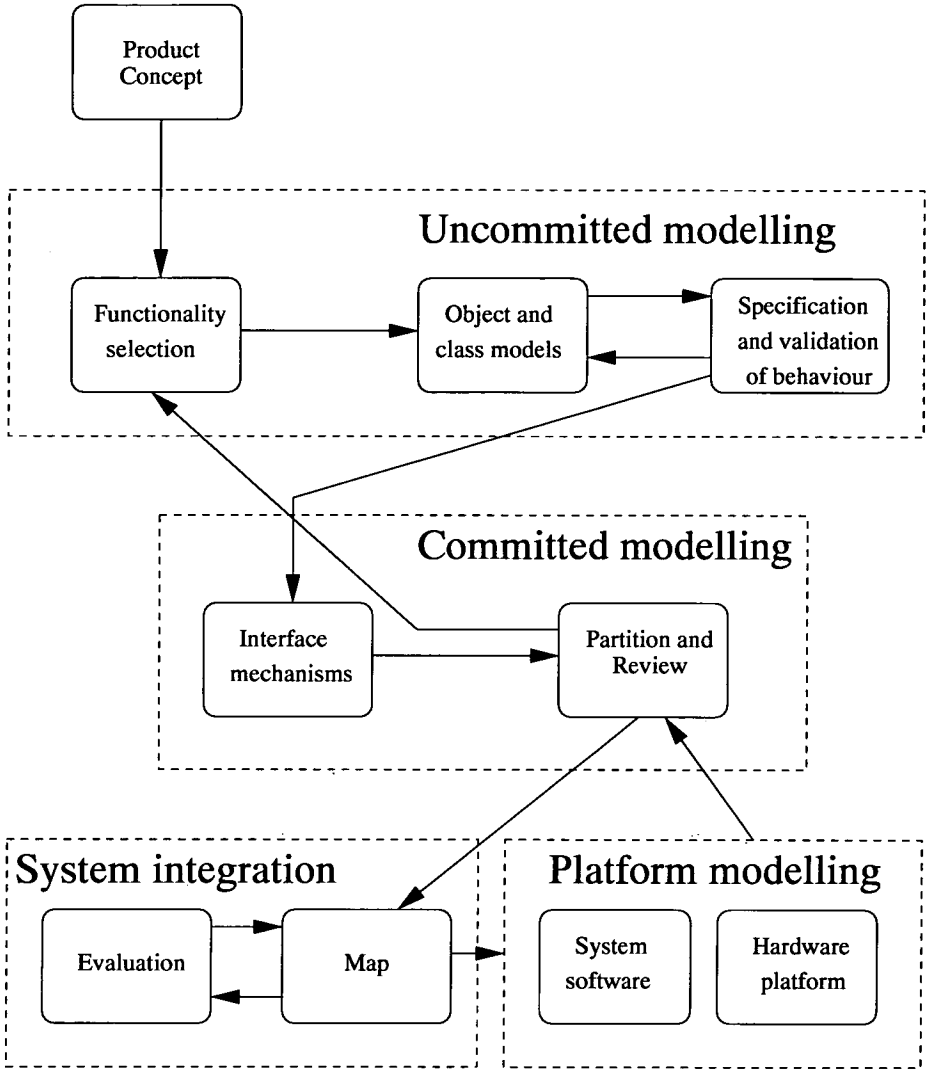

Figure 3.1: *HASoC lifecycle [2]*

3.6.2.2 Uncommitted Modelling

This stage is divided into three main phases, functionality selection, object and class models and specification and validation of behaviour.

In the first phase, functional requirements are analysed and the overall task of developing the system to meet these requirements is broken down. The core functionality of the system is developed and this core functionality will be application specific. In the AOI system for example the detection of orientation and displacement of Integrated Circuits would be a core activity. A use-case-driven approach is one method of exposing the required functionality of the system.

This approach is strongly supported by UML.

In the objects and class models phase, object and class models are developed for the selected use cases using the notation of UML for real-time [9]. A typical approach to accomplish this task would be to develop sequence diagrams for key use case scenarios, and then synthesise the class and capsule collaboration diagrams from the sequence diagram.

In the specification and validation of behaviour phase, in order to provide a finer degree of behavioral validation, detailed code would be added to the skeleton executable model. This code could be in C++ for example.

3.6.2.3 Committed Modelling

The commitment of a model concerns moving an abstract executable model towards an implementable specification. The stage is concerned with partitioning the model into hardware and software implementations, and then allocating the resulting 'committed capsules' to the processing elements in the platform.

The committed modelling stage is divided into two main stages. In the first stage, interface mechanisms, the external interface is considered. In some projects the mechanisms by which the environment interacts with the system will be fixed at the start of development, whereas in others they must be determined as part of the design process. In the AOI system development the mechanisms are fixed in terms of the system interacting with the environment through the use of a camera for taking images of printed circuit boards. The other mechanism for interacting with the user is a standard PC, keyboard, mouse set up with a GUI interface. Therefore, there are no specialised hardware requirements in terms of graphic displays or input devices for this project.

In second stage of committed modelling, partition and review, partitioning is performed at the object level. Some objects are identified for software implementation while others are implemented in hardware and in HASoC this is aided by the availability of an executable model. In the AOI project all objects are targeted for execution on programmable cores for maximum flexibility. Section 2.10.2 gives detailed reasons why the use of programmable cores gives optimal flexibility and enhanced performance.

3.6.2.4 Platform Modelling

The platform model consists of a hardware platform to execute the committed model and system software support. The platform model is therefore concerned with processors, buses, memories, RTOS and device drivers.

The hardware platform is an architecture that satisfies the hardware and software constraints of the system. A pre-existing platform may be used for the implementation of the system. It is more common to customise or reconfigure a system with addition, removal, replacement or minor modification of hardware and software objects.

In the AOI project the use of the ARM Integrator platform [85] provides maximum flexibility for reconfiguring the hardware platform for a system. For the AOI project an ARM Integrator CM7TDMI [86] was selected to run on the ARM Integrator platform. The CM7TDMI incorporates an ARM7 processor core [80]. Prototype application software can be modified and executed on the ARM7 processor with a fast code, test and debug cycle. Modifications to increase the processing power of the platform are possible, by employing a higher specification ARM core module [87], or purchasing further CM7TDMI modules which can run in parallel. The ARM Integrator platform is designed for easy expansion with several processor cores running in parallel if necessary.

In the AOI project the ARM platform was enhanced further with the addition of a Texas Instruments EVM6201 DSP to meet the performance requirements of the system. The application software was then profiled and partitioned across this enhanced platform with the DSP executing the computationally expensive parts of the software.

The partitioning of the application software between the ARM processor and a co-processor is similar to the approach adopted in [88]. In this paper an embedded software implementation of an Adaptive Differential Pulse Code Modulation (ADPCM) algorithm is developed which is a speech coding algorithm. Through code optimisations significant improvements in performance and code size are achieved. This was achieved through taking advantage of the characteristics of the specific ARM processor, for example its instruction set and number of general purpose registers. Through hardware and software partitioning further improvements in speed and code size are achieved. Similar to the AOI project this was achieved through evaluating the complexity of different blocks of the ADPCM algorithm. The approach differs from the AOI project in that a dedicated co-processor is designed rather than using a standard

Texas DSP. This dedicated co-processor therefore implements computationally expensive tasks directly in hardware.

For this work the application software went through several revisions or prototypes. Software objects were added and replaced to meet the functional and non-functional constraints of the system. It was therefore essential that a software architecture was developed that could be easily extended. The addition of components should not entail extensive changes to existing components. This was achieved through encapsulation and modularisation [89] of the software in C and C++. The HASoC approach directly supports this methodology.

The platform model is concerned with describing the overall execution environment in sufficient detail to facilitate the implementation of the complete system. There are a number of complexities to this approach. They include the direct communication between software and hardware objects in the committed model being accomplished via the services of the platform model. Further, the platform model shows the hardware architecture of the system, which includes the supporting hardware as well as the fixed-function hardware IP cores. For these reasons the platform model has two major elements: the Software-Hardware Interface Model (SHIM) and the Hardware Architecture Model (HAM). The detailed composition of these two models are described in [2].

3.6.2.5 System Integration

The aim of this stage is to execute the committed model on the hardware architecture model. An evaluation process is carried out which is a performance assessment of the system. This process determines whether or not the current platform can execute the committed model in a functional sense as well as satisfying design constraints. If necessary the result of this process will be modifications to the hardware platform. Modifications to the committed model, in terms of allocation of objects to hardware and software, may also be required so that constraints are satisfied.

3.6.3 HASoC and AOI: conclusions

HASoC is based on iterative system construction which is more responsive to changing requirements. Work can commence on the underlying platform much earlier in the lifecycle. This can proceed concurrently with iterative development of the application oriented execut-

able model. There is significant concurrency in the lifecycle which is important because it enables developers to work in parallel, but always with reference to a homogeneous model of the complete system.

HASoC is directly relevant to this work because the AOI development is iterative with successive prototypes being developed to meet functional and non-functional requirements. The HASoC methodology could therefore be applied to further prototypes of the AOI system particularly when a customised SOC platform is developed. This platform can be developed in parallel with further software development and the methodology directly supports this approach.

HASoC supports the use of a re-useable library of software components which have been developed for the AOI system. This software library is targeted at standard programmable cores. The programmable cores can be replaced with more powerful architectures with minimal changes to the software. Through execution time analysis against system constraints, software components can be replaced with optimised versions. To avoid significant changes to other software components fixed interfaces have been used with just the implementation of the components being modified to improve performance. This approach is directly supported by HASoC.

3.7 System-On-Chip Technology and Vision Systems

The targeting of machine vision algorithms for SOC for an AOI application is novel. Algorithms for many machine vision applications are too computationally expensive for even state-of-the-art embedded platforms. Therefore, in this work the complexity of the vision algorithms and the platform on which they run are central issues.

There is work published in the area of application specific vision chips for computationally demanding applications. In [90] a smart vision system is presented for real-time vision applications. The system integrates a camera with a programmable neural computer and an advanced microcomputer onto an SOC achieving one tera-operation-per-second performance, making it suitable for many commercial applications.

In [91] an application specific vision chip is presented which combines analogue and digital processing with applications to automotive image processing, eye tracking and visual inspection. The analogue part implements computationally intensive operations on a massively par-

allel array, with the remaining operations implemented with a reduced complexity dedicated digital processor. These analogue and digital components are integrated onto an SOC.

In [92] an embedded vision system called EyeQ for automotive applications for accident reduction and driver assistance is described. The system uses two 32-bit RISC ARM946E CPUs and four Vision Computing Engines (VCE), a multichannel DMA and several peripherals. The four VCEs and the ARM946E perform all the intensive vision computations required by the applications such as tracking and pattern classifications.

The objective of the work in [93] is to investigate the use of a reconfigurable computer system targeted for real-time computer vision applications. The hardware architecture for the system is based on a FPGA. Reconfigurable systems combine a reconfigurable hardware processing unit with a software programmable processor. These systems allow the customisation of the hardware in order to meet the computational requirements of different applications. The use of a FPGA adds design flexibility and adaptability. This contrasts to this work where flexibility is gained through using standard programmable cores which give a fast design, code, test cycle for machine vision algorithms.

3.8 Summary

In this Chapter the field of real-time embedded systems has been defined. The different types of embedded system are detailed. The system presented in this work is classified as a soft real-time system where the missing of deadlines for the software leads to degradation in system performance rather than system failure.

This Chapter defines the important field of Software/Hardware Co-design. The Chapter discusses examples of applications where mapping of an application into programmable and dedicated components has given significant performance benefits.

The Chapter goes on to define SOC technology, the technology used for this work. SOC technology was selected because of the significant performance benefits for running embedded software. The additional demands on embedded software development through using this type of technology are detailed with reference to the AOI system development. The key issue in embedded software development for SOC platforms is component re-use. Attention to the issue of re-use is essential for meeting the demands of the market place where increasingly short

timescales and high product functionality and reliability are the norm.

The Chapter goes on to detail why system level design methodologies are important in the development process. These methodologies are essential for SOC development. These methodologies also support re-use. This area is illustrated through a detailed analysis of the HASoC methodology with reference to the AOI development.

The Chapter concludes with detail on existing literature on machine vision systems which utilise SOC technology. It should be emphasised that the application of complex image processing algorithms to an AOI system based on SOC technology is novel.

Chapter 4
Image Detection using Reduced Complexity Hough Transforms

4.1 Introduction

This Chapter presents object recognition techniques based on the Hough Transform, as defined in [25] and [94], for detecting Integrated Circuits within a variety of Printed Circuit Boards images autonomously and without the need to be assisted by CAD data. The Printed Circuit Boards used are characterised by being densely populated and contain many features other than Integrated Circuits and therefore represent a difficult object recognition task.

Image recognition is an important part of the Machine Vision field. Object detection is a important class of problem within Image recognition. Traditional approaches of object recognition include the use of Hough Transforms [94]. Methods of accelerating the traditional Hough Transform have been proposed. These methods include rectangular image decomposition [95], hierarchized schemes [96], fast linear hough transforms [97] and fast hough transforms from grey scale image [98].

For the technique by Gatos et al [95] an image is decomposed using rectangular blocks and the contribution of each whole block to the Hough transform space is evaluated and this is in contrast to evaluating every pixel. The technique by Guil et al [96] enhances fast hough transform techniques and solves problems such as erroneous solutions, point redundance, scaling, and detection of straight lines of different sizes. The authors also explore the possibility of a parallel algorithm for multiprocessors. The algorithm can be adapted for detection of shapes as well as lines such as circles and ellipses. A third technique is proposed in Vuillemin [97] and this approach is based on a recursive algorithm for raster-scan line drawing. The technique has a divide and conquer recursive structure similar to that of the classical Fast Fourier Transform (FFT). Guil et al [96] provide a breakdown of the various schemes and how they differ.

This chapter presents the use of Hough Transforms for detecting Integrated Circuits (ICs) within images of a Printed Circuit Boards (PCBs). A sample test image with five ICs is shown in Figure

4.1. The ICs are of varying size and represent a difficult object recognition problem because of features such as chip legs, tracks and Chip IDs.

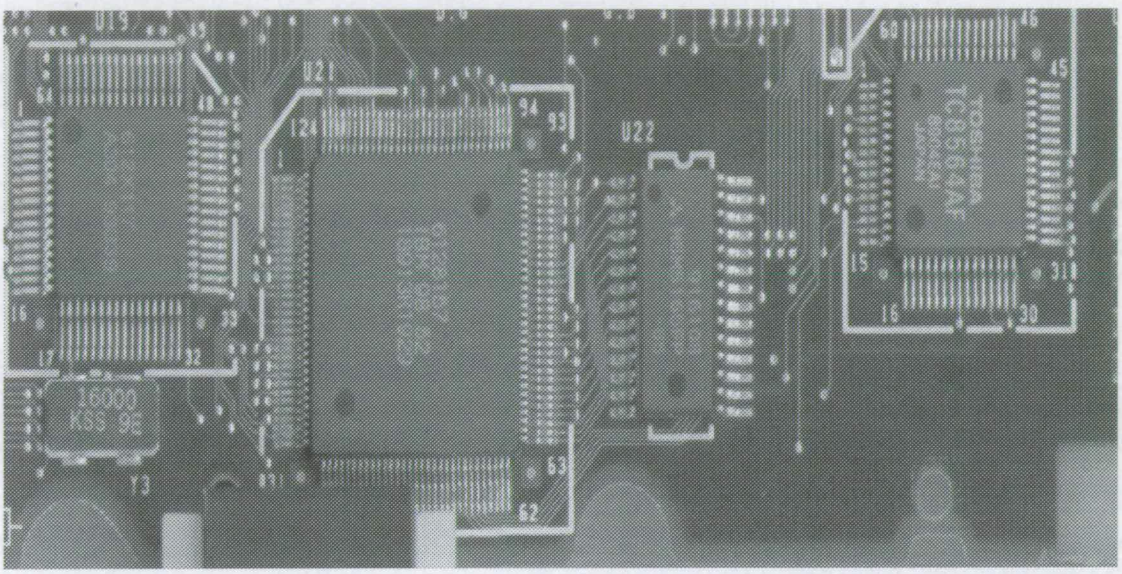

Figure 4.1: *Sample Grey Scale Image*

Through detecting the positions of the ICs in a reference image it is then possible to detect whether further samples have the ICs present. The IC position information can then be used for further processing such as checking the orientation and placement position of the chips.

The chapter presents implementations of the Hough Transforms for detecting lines and rectangles. Due to the high throughput of samples under test the efficiency of the IC detection algorithms are an important issue. The Hough Transform techniques are found to be too computationally complex and memory hungry for complex images.

For the techniques presented in this chapter, the SUSAN operator [33] edge detection operator has been used to derive an edge map. A sample edge map containing three ICs is given in Figure 4.2. The SUSAN operator is more efficient than competing techniques such as the Canny Edge [34] detector which makes it suitable for embedded applications such as AOI. All the techniques presented in this chapter are designed to work without the use of CAD data. This is an important feature as board CAD data may not be available in a manufacturing environment where board modifications may be frequent and retest cycle may be short.

The chapter is organised as follows: in Section 4.2 presents an implementation of the Hough

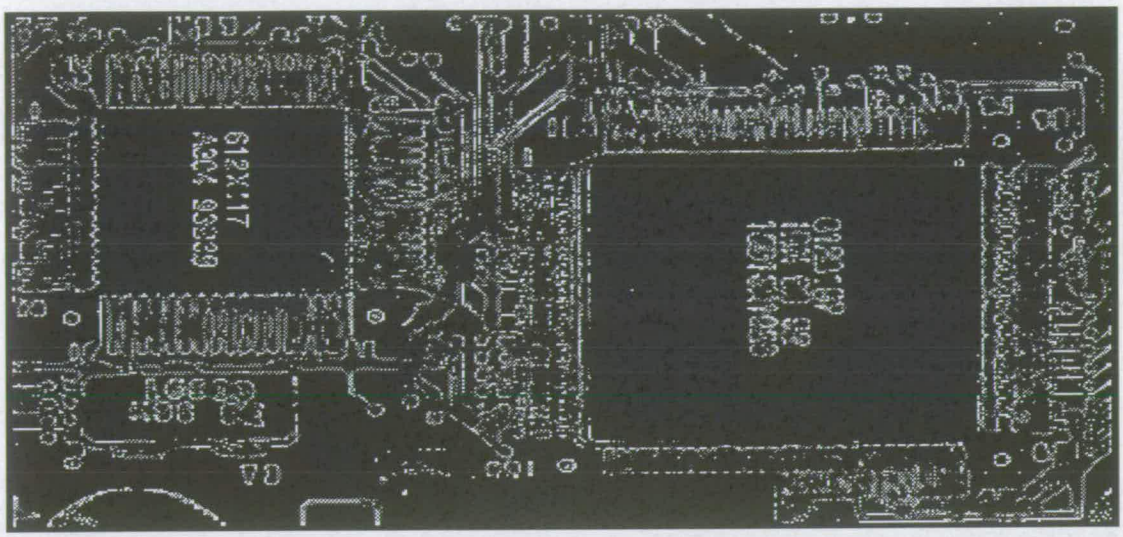

Figure 4.2: *Edge Map result from application of the SUSAN operator*

Transform for detecting rectangles and this implementation is too computationally complex and memory hungry; in Section 4.3 presents a novel technique which uses Hough Transforms for detecting lines in an edge map, this technique only produces correct IC position information for simple images and is inefficient for noisy multiple IC images; in Section 4.4 presents some results from the work; and Section 4.5 presents some conclusions.

4.2 Original Hough Transform for Detecting Rectangles

The Hough Transform is a method of object recognition and is a well known segmentation technique. The Hough Transform in it's original form is discussed in [94]. Methods of accelerating the Hough Transform are discussed in [95], [96] and [97].

A Hough Transform based on the original form has been developed for detecting Integrated Circuits within an edge map. Due to the computational complexity of this technique and memory requirements this technique has not been verified on test samples.

The Hough Transform technique scans the image and for each pixel votes into accumulator space. The number of potential ICs for each pixel point is very large. The accumulator array is four dimensional with two dimensions for the start position of the IC and two dimensions for the width and height of the IC. This technique proceeds as in Pseudo code 2.

There are four main cases for each pixel point each of which is similar in it's processing. Case 1 has been expanded in full. For this case we assume the point is on the right of the IC. Given this point the accumulator array is then incremented according to all possible ICs. This processing is carried out in the for loops with variables nPX and nPY searching from the origin to the pixel position. Within these loops nPW and nPH are set to give the width and height of the IC. These variables are set with reference to the maximum width and height of the IC set at the start of the algorithm.

4.3 Hough Transform Reduced Complexity Technique

An alternative technique to a standard Hough transform for rectangles has been implemented. With this technique a Hough Transform is used to detect lines. From this list of lines processing is carried out to find the intersection points between pairs of lines. This involves checks to see if lines are parallel, and therefore can't have an intersection, and checks to see whether the lines are perpendicular. We are only interested in cross points of lines perpendicular to each other as these are potential corners of integrated circuits.

The lines are represented parametrically. Pseudo code 3 is an example of finding crosspoints between a pair of lines.

From the sets of x and y coordinate crosspoints a cross product set is formed. From the cross product set an evidence matrix is computed which represents potential start and end points of integrated circuits. The whole image is then processed with each pixel point voting into the evidence matrix, voting for start and end pairs according to whether the pixel is in the region defined by the start and end points. An evidence list is then computed with the start and end pairs stored according to the evidence for their presence in the image. Potential start and end points of IC's can then be retrieved from the evidence list according to the evidence of presence in the image.

An algorithm for the procedure is as Pseudo code 4.

The algorithm starts with the input of the edge map to be processed and then performs the Hough transform for detecting lines. The main **Do..While** loop compares each potential line with every other potential line and computes intersection points. These points are stored in a intersection list. The cross product and the evidence matrix are computed from the intersection

list. The second main loop is the voting procedure into the evidence matrix. This voting procedure is carried out for every pixel in the image. From the evidence matrix an evidence measure is computed and stored in an evidence list according to the number of votes registered in the voting procedure. The rectangle start and end points are then output according to the evidence for the rectangle in the image.

4.4 Results

Hough Transform Based Techniques

For the Hough transform presented in Section 4 the number of operations for the image in Figure 4.2 which contains 16410 pixels is 10.4 x 10^9. The transform is therefore computationally complex for multiple IC images. The execution time of the Hough transform is of order O(n^6) where n is the number of pixels in the image. This complexity measure was found through analysing the number of source code operations carried out for a test image.

For the Hough Transform reduced complexity technique presented in Section 4.3 applied to the image in Figure 4.3 which has two ICs the algorithm outputs the correct positions for the ICs . Note all chip ID and non-chip edge information has been removed from this image using an image processor before applying the algorithm.

The following are outputs from the algorithm giving the correct x and y co-ordinate positions of the IC's.

x1: 417 x2: 522 y1: 178 y2: 21
x1: 21 x2: 277 y1: 206 y2: 5

Even with image pre-processing the edges of the ICs are not well defined. This results in there being many lines registered by the Hough Transform and there being many crosspoints. Running on a Sun Ultra 60 under Unix on the image in Fig 4.3 the IC detection process took 6.32 minutes which is clearly far too long for an embedded system which will have to process an entire image in one or two minutes.

This technique is more computationally expensive for complex images with many ICs where the Hough Transform registers lines which are not part ICs edges. The number of potential crosspoints is therefore large. The voting procedure for every pixel into the matrix of potential

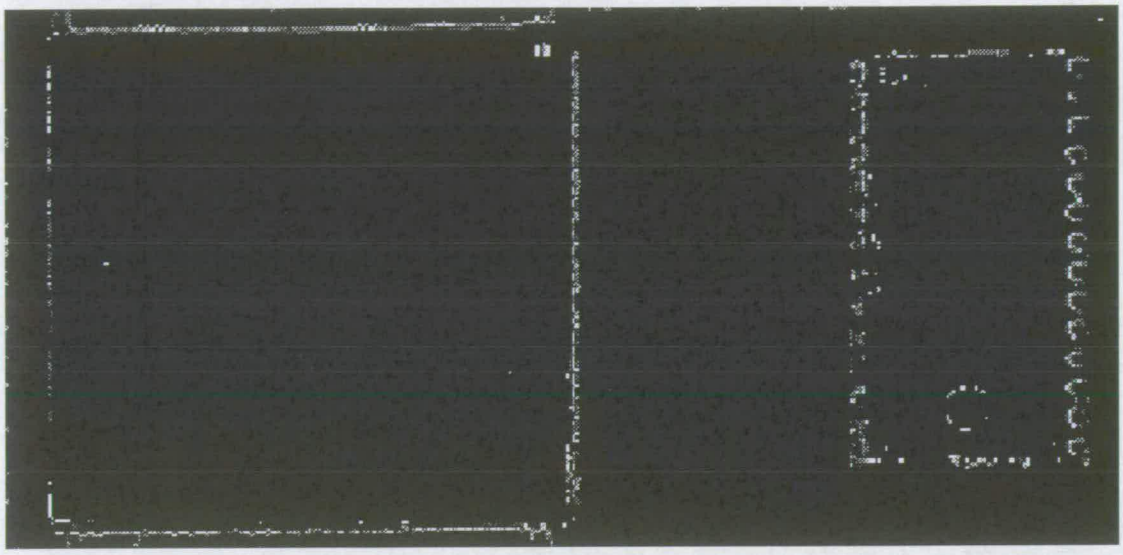

Figure 4.3: *Two ICs after image pre-processing*

start and end points becomes computationally complex.

Results of Methods of Removing Chip ID

To overcome the problem of Chip IDs registering as lines noise reduction techniques were applied. The reduction techniques looked for blocks of pixels and removed part of the Chip ID. The problem with this technique is the removal of vital information from the images e.g. edges of ICs. Figure 4.4 is a single IC image.

Figure 4.5 shows Figure 4.4 with noise reduction applied. Some of the Chip ID has been removed but so has the right hand edge.

The morphological operators Close was applied which is Dilation followed by Erosion as discussed in [7] giving Figure 4.6. Noise reduction techniques were then applied to the resultant binary image to remove the Chip ID giving Figure 4.7. This was only partially successful with part of the edge information still removed from the IC.

Figure 4.4: *Single IC image*

Figure 4.5: *Single IC image with noise reduction*

```
// Initialize minimum and maximum width and height of IC we are trying to detect
Initialize nMinWin , nMinHgt , nMaxWid , nMaxHgt


Loop x for zero to image width
        Loop y for zero to image height

        If pixel at position (x,y) in image

        // Case 1: Assume point on right of Integrated Circuit
        // increment accumulator array according to all possible IC's
        Loop for nPX zero to pixel position x
                Loop for nPY zero to pixel position y
                nPW = x - nPX
                If nPW < nMaxWid and nPW >= nMinWid
                        If y - nPY > nMinHgt
                                nPH = y - nPY
                        Else
                                nPH = nMinHgt
                        Endif
                        Do
                                increment accum[nPX][nPY][nPW - nMinWid][nPH - nMinHgt]
                                increment nPH
                        While nPH < nMaxHgt - nPY and nPH < image height - nPY
                Endif
                End Loop
        End Loop

        // Similar process for the following cases

        //Case 2: Assume point on left of integrated circuit
        // increment accumulator array according to all possible IC's
        //Case 3: Assume point on top of integrated circuit
        // increment accumulator array according to all possible IC's
        //Case 4: Assume point on bottom of integrated circuit
        // increment accumulator array according to all possible IC's
        Endif
        End loop
End loop
```

Pseudo code 2: *Hough Transform Algorithm for Detecting Rectangles*

x cos q + y sin q = s **(1)**
x cos a + y sin a = t **(2)**

To find intersection points x and y we have to derive expressions for x and y in terms of q and a and s and t. A method of Gaussian elimination [99] is used as follows:-

multiply **1** by cos a / cos q

=> x cos a + y cos a / cos q * sin q = cos a / cos q * s **(3)**

then subtract **3** minus **2**

=> y * (cos a / cos q * sin q - sin a) = cos a / cos q * s - t **(4)**

re-writing to get an expression for y

=> y = (cos a / cos q * s - t) / (cos a / cos q * sin q - sin a) **(5)**

to get an expressions for x substitute **5** into **1** or **2**

substituting into **2**

=> x cos a + ((cos a * s / cos q - t) / (cos a * sin q / cos q - sin a) * sin a) = t **(6)**

re-writing to get an expression for x

=> x = (t - ((cos a * s / cos q - t) / (cos a * sin q / cos q - sin a) * sin a)) / cos a **(7)**

by entering expressions 5 and 7 into a tool such as xmaple they can be simplified to the following:-

y = (s * cos a - t * cos q) / (sin q * cos a - sin a * cos q) **(8)**

x = - (s * sin a - t * sin q) / (sin q * cos a - cos q * sin a)**(9)**

Pseudo code 3: *Finding Crosspoints between pairs of lines*

```
Input edge map
Perform Hough transform to find lines
Search through hough space and form line list representing potential lines in the image

// find intersections between potential lines
// we want to look at the first line and compare it to all other lines
// then look at the second line and compare it to all other lines

Line List pointer 1 = First element in line list
Line List pointer 2 = Second element in line list
Do
    If not first time through the loop
        Line List pointer 1 = Next element from line list
        Line List pointer 2 = First element from line list

    While Line list pointer 2 is not at end of list
        Check to see if lines are parallel by computing gradients

        If lines are not parallel
          Check to see if lines are perpendicular
          If lines are perpendicular
                Find intersection
                Add intersection co-ordinates to intersection list
          End if
        End if
        Line List Pointer 2 = Next element of line list
    End while
End while

Form cross product of intersection points from intersection list
Compute evidence matrix
Loop x for zero to input image width
    Loop y for zero to input image height
        If input image(x , y) is a pixel
          Vote for rectangle in evidence matrix
    End loop
End loop

Compute evidence measure from evidence matrix votes
Create evidence list according to number of votes for each rectangle start and end point
Output rectangle start and end points according to the evidence for their presence in the image
```

Pseudo code 4: *Hough Transform reduced complexity technique*

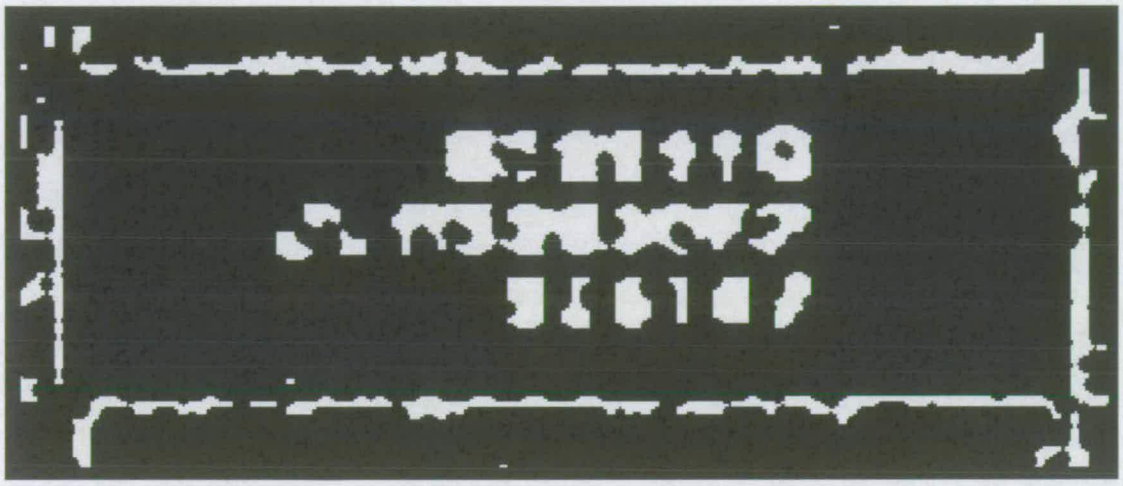

Figure 4.6: *Result of binary Close operator*

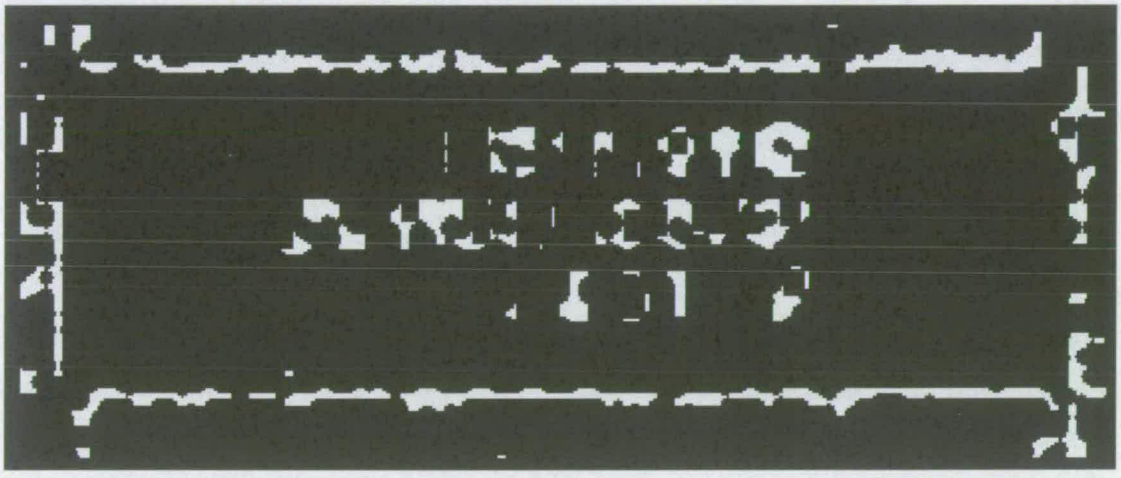

Figure 4.7: *Noise reduction technique applied to result of Close operator*

4.5 Conclusions

This chapter has shown that the use of the original Hough transform for the detection of integrated circuits within images of PCBs is too computationaly complex for the application presented in this work. The Hough transform is found to be of order $O(n^6)$. In this chapter a reduced complexity technique based on a hough transform for detecting lines and then a procedure for detecting rectangles is also developed. This technique was not found to be efficient

enough for the samples under test. The failure of this further technique was due to the Chip IDs in the centre of the chips registering as lines. The final techniques presented in this Chapter included the use of noise reduction techniques and morphological operators to remove the Chip ID. These techniques were only partially successful, with important edge information being removed from the image as well as Chip ID.

Chapter 5
Enhanced Image Detection on an ARM based Embedded System

5.1 Introduction

This Chapter takes a different approach to object recognition than Chapter 4 which used Hough Transforms. Having found that Hough Transforms are too complex, this Chapter describes low complexity object recognition techniques based on region growing.

This Chapter presents a new technique for the detection of Integrated Circuits within images of Printed Circuit Boards autonomously and without the need to be assisted by CAD data. The technique is a key part of a suite of algorithms targeted for an embedded SOC architecture based on the ARM7 platform for real-time detection of PCB images for diagnostic purposes. The technique has a significant reduction in complexity when compared to conventional approaches such as the Hough Transform. The reduction in complexity makes the approach ideal for an embedded vision application such as the one described in this paper. This chapter presents the technique, the target embedded architecture and results showing the reduction in complexity when compared to a Hough Transform.

Image recognition is an important part of the Machine Vision field. Object detection is an important class of problem within image recognition. Object recognition and manipulation algorithms are characterised by being computationally complex due to the size of both image and source system added to the large number of complex arithmetic operations. It is extremely desirable that such applications are performed on a standard SOC embedded processor without the need for large and expensive memories and co-processors. The main aim of this Chapter is therefore to show the feasibility of implementing object recognition algorithms on a standard SOC target which provide the advantage of flexibility in addition to that of real-time speed.

Effective methods of object detection of Integrated Circuits (ICs) are of considerable interest to developers of Automatic Optical Inspection (AOI) systems for analysing Printed Circuit Boards (PCBs). AOI systems are becoming more important in the manufacturing process as the

complexity of electronic circuits and there associated testing requirements increases. Existing work on AOI systems for PCB analysis is detailed in [66] and [100]. In [66] the breaks in wires and short circuits are detected and the technique used is referential matching between the stored reference image and the captured test image. In [100] novel structured-lighting inspection technology is used to implement an efficient solution to the detection of solder joint problems. This Chapter describes the development of a low complexity image recognition technique for an AOI system targeting an embedded SOC system based on the ARM7 target processor. For this system real-time speed is of prime concern.

The ICs in the image are of varying size and represent a difficult object recognition problem because of features such as chip legs, tracks and chip IDs. Through detecting the positions of the ICs in a reference image it is then possible to detect whether further samples have the ICs present. The IC position information can then be used for further processing such as checking the orientation and placement position of the chips.

Due to the high throughput of samples under test, the complexity of the algorithms for analysing the PCBs is an important issue. Reduced processing time for analysis of PCBs can be achieved through execution of the algorithms on an embedded SOC processor such as the ARM7.

Traditional approaches to image detection include the use of Hough Transforms [94] but these techniques have a high computational complexity and memory requirements, as shown in Chapter 4. Methods of accelerating the traditional Hough Transform have been proposed, including parallel implementation [101], hierarchical schemes [96] and application dependent fast algorithms [102]. The work in Chapter 4 includes an implementation and analysis of a Hough Transform based techniques for detecting lines and then further processing to detect rectangles corresponding to ICs. This technique fails to produce good results because the edges of the chips are ill defined in the edge map. A further problem is the existence of other features in the image such as chip IDs which register as lines.

This Chapter presents a new technique for the detection of ICs within images of PCBs which uses region growing [103]. The application of region growing to the detection of ICs is a novel approach. The limitations of this technique are shown below. The technique uses multi-thresholding with the limitation that ICs only in a narrow band of intensities are detected. ICs may be in widely varying bands of intensity, through the use of metallic or dark grey casing for example. The Chapter goes on to describe two techniques for getting around this limitation. The

first technique is based on segmentation and uses the output of the SUSAN operator to segment the original image. The second technique uses a Gaussian probability distribution derived from processing a Region Of Interest (ROI). The intensity of the pixels in the computed probability image is then proportional to the probability function.

Conventional IC detection systems are assisted by CAD data, such as placement or connectivity information to improve diagnostic performance. However in a number of environments it is not possible to obtain such data. Examples would be an embedded environment where design is continually changing, such as the one described here.

This chapter is organised as follows: in Section 5.2 presents the algorithm based on region growing for efficient detection of Integrated Circuits; in Section 5.3 details the target ARM platform; in Section 5.4 gives results from the implementation and analyses the complexity of the algorithms compared to the Hough Transform presented in Chapter 4; in Section 5.5 gives algorithms for two methods of pre-processing of images of PCBs where the Integrated Circuits are in widely varying bands of intensity; and Section 5.6 draws some conclusions from this Chapter.

5.2 The Algorithm

This technique proceeds as follows: Firstly edges are detected from the grey scale image using the SUSAN operator [33]. A binary edge map is then generated through applying a multi-threshold to the grey scale image. Approximate chip centres are then located in the binary edge map using a region growing technique to segment the image and heuristics to identify regions likely to be textual chip ID information. The heuristics exploit the fact that chip ID regions are located closely together in the image. Segmented text is then removed from the original edge image by applying chip ID information as a mask. Region growing is applied to the resulting edge map using the approximate chip centres as seed points to find the IC start and end co-ordinates.

The technique operates without the use of CAD data for the PCB. If CAD data is available then the calculation of chip centroid positions is not necessary. However the chip ID information still has to be removed for the final region growing step to work.

The SUSAN operator has been selected because of the marked performance over competing

techniques such as the Canny Edge [34] detector which makes it suitable for an embedded application such as AOI.

The algorithm for the process is shown in Pseudo code 5 with a flow diagram in Figure 5.1.

This algorithm begins with the application of the SUSAN operator to the grey scale image. A multi-threshold is then applied to the Grey Scale image to give a binary inverse image. The main loop then searches the whole inverse image and grows regions where the intensity is zero. If the number of pixels in the region is within set bounds then the region start point is recorded in a region list. These bounds are set as we are only interested in chip ID regions which are in a limited range of total number of pixels. This region list is then processed to remove regions which are not parts of chip IDs. This is carried out through using a heuristic that chip IDs form regions which are grouped close together in the image. This linked list processing with dynamic data structures is shown in Figure 5.2. Dynamic data structures are necessary due to the number of chip ID regions in the image being unknown at the start of the processing.

Figure 5.2 shows that first a list of all the regions are built, with the X and Y start positions of the regions being recorded. In Figure 5.2 more complex data structure is then constructed, for each potential IC centre there is a sublist containing all the regions local to that IC centre. The approximate chip centre is then computed from the average of all region locations in the sublist.

A region growing step is then applied to the inverse image from the start positions in the region list to give a chip ID map. An example chip ID map derived from this process is given in Figure 5.4. The approximate chip centroid positions can then be calculated from the start points of the regions in the region list. The chip ID map is then applied as a mask to the edge map to remove the chip IDs. An example edge map with the chip IDs removed is given in Figure 5.3. A further region growing step can then be applied to find the start and end positions of the ICs. Without the removal of the chip ID this stage would fail. The edges of the ICs can then be highlighted using the IC start and end position information. Results of this process are shown in Figure 5.5, Figure 5.6 and Figure 5.7.

```
Input algorithm parameters
        minimum region size , maximum region size
        lower intensity threshold , upper intensity threshold
        Chip ID search index

Apply SUSAN operator to grey scale image to give edge map

/* segment chip area from rest of image */
Apply upper and lower intensity threshold band to grey scale image to give binary inverse image

/* region growing searches from a seed points into areas of zero intensity */
/* in the binary inverse image - only record regions within maximum and minimum region size */
Loop for 1 to inverse image width
        Loop for 1 to inverse image height
                For each region start point grow region
                        Record number of pixels in each region
                        If number of pixels > minimum region size
                        And number of pixels < maximum region size
                            Record region start point in region list
        End loop
End loop

/* use Chip ID search index - utilise heuristic that Chip ID regions are grouped closely together */
Process region list
        Remove regions from region list which are not part of Chip IDs

/* now produce Chip ID map from region list */
Loop for 1 to end of region list
        For each region grow region in inverse image to produce Chip ID map

Calculate chip approximate centroid positions from start points of regions in region list

/* delete Chip ID from edge map */
Apply Chip ID map to edge map to give processed edge map

/* now perform final region growing step to find edges of ICs */
Loop for one to number of approximate IC centres
        Search from approximate centre of ICs until edges of ICs found

For each IC found in edge map
        Draw lines showing IC edge utilising IC Start and End point information

Output edge map with ICs highlighted
```

Pseudo code 5: *Algorithm for IC detection process*

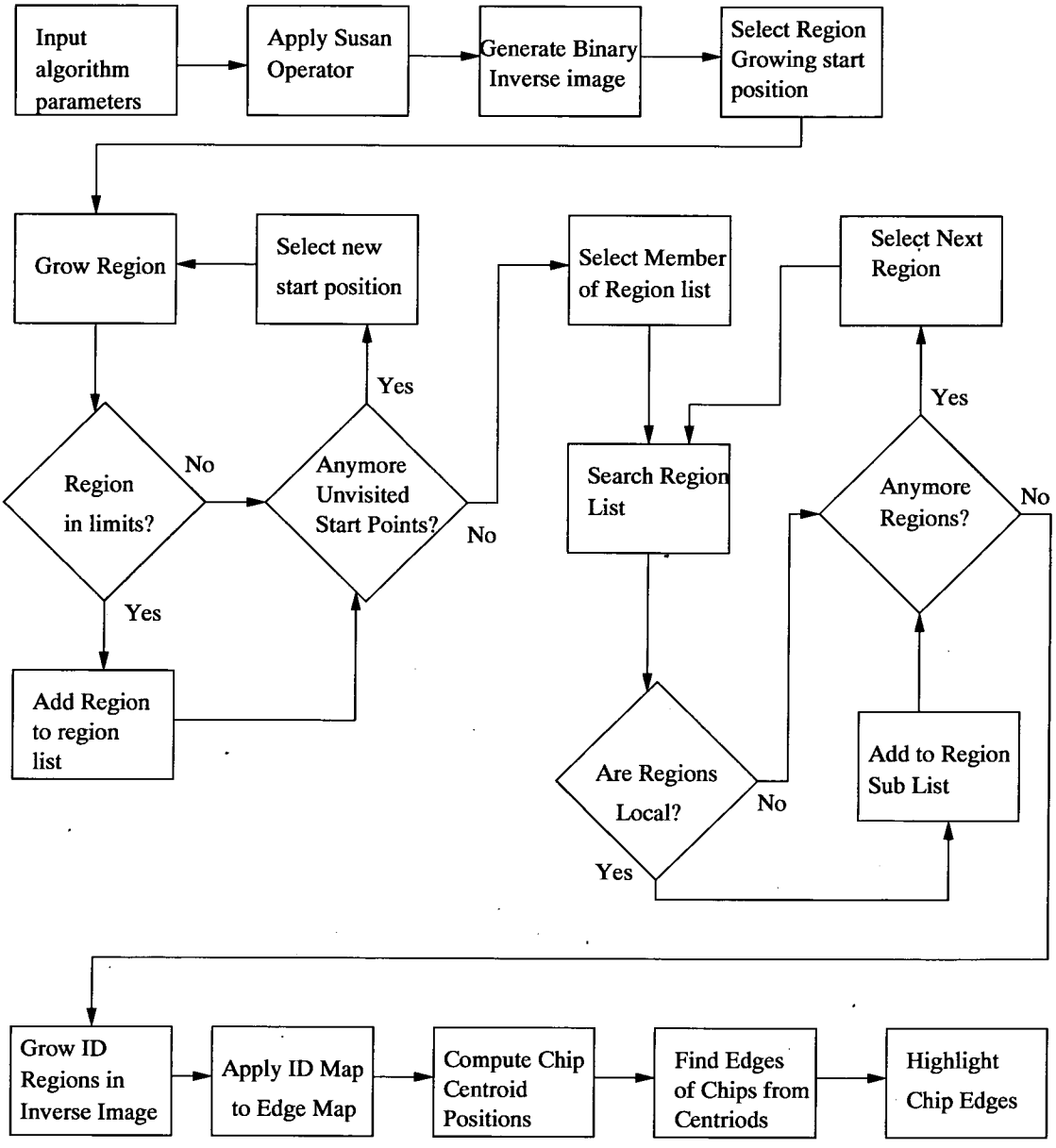

Figure 5.1: *Flow Diagram for Region Growing Process*

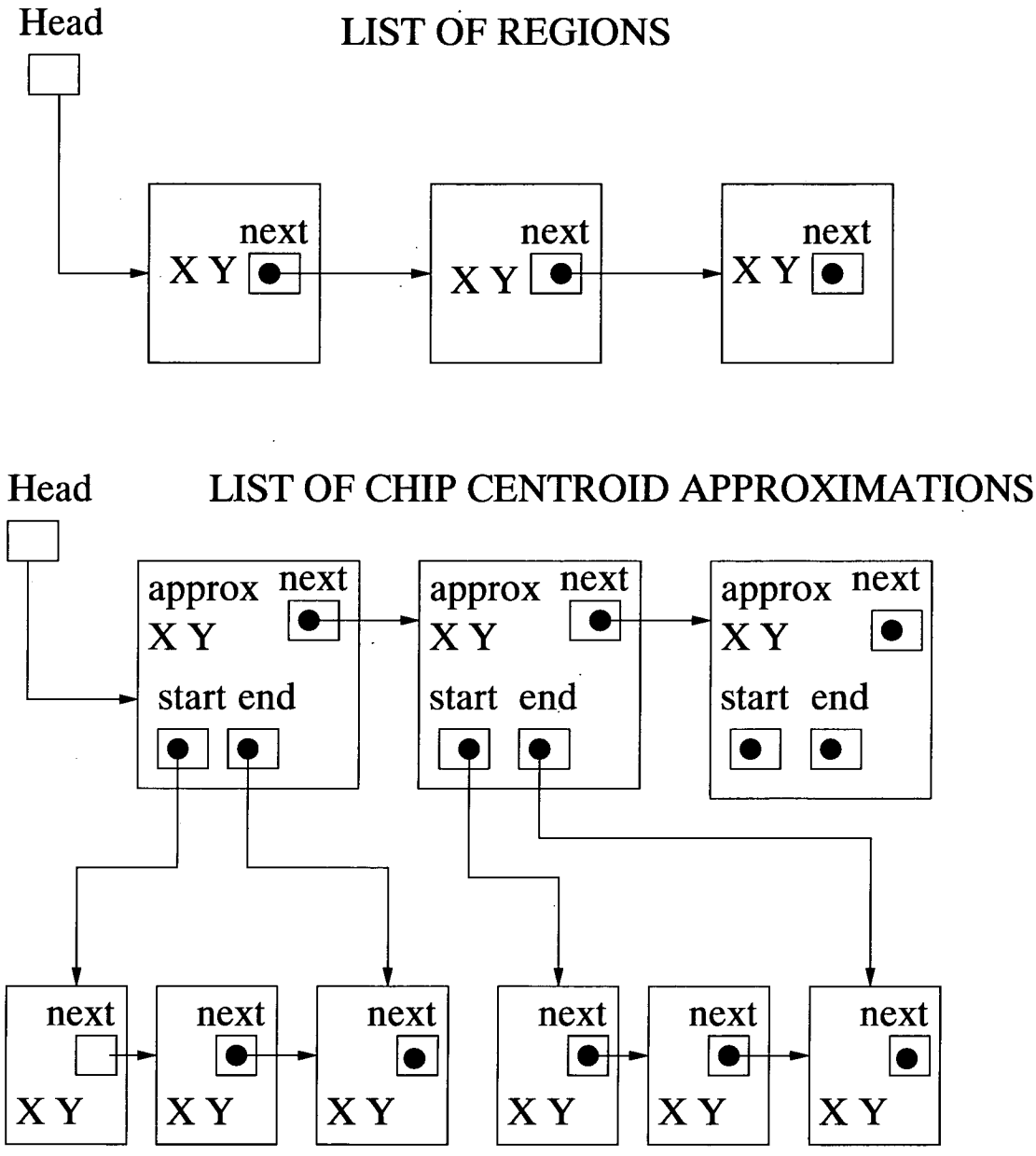

Figure 5.2: *Region Growing Linked Lists*

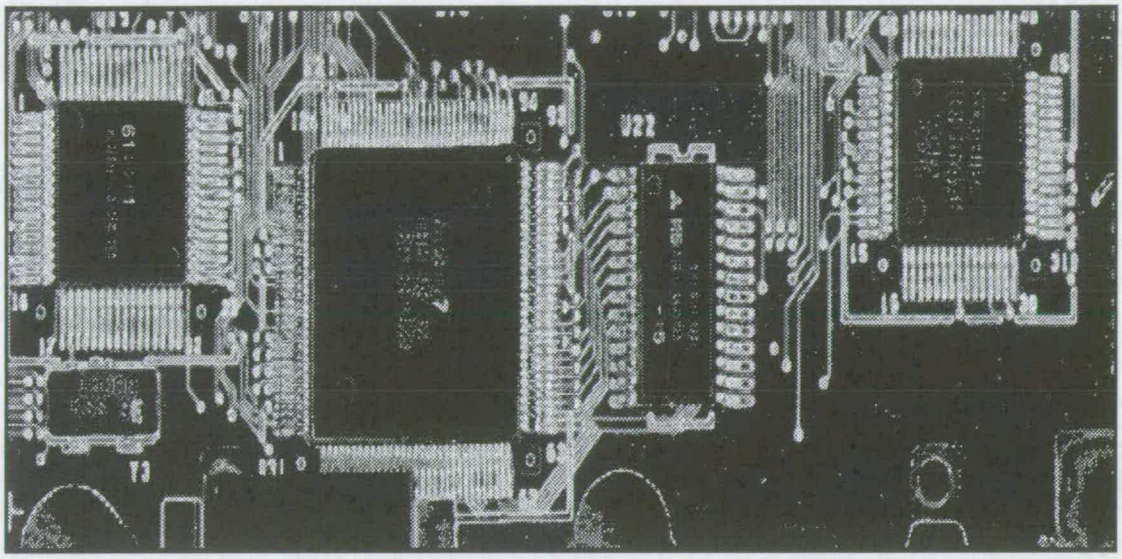

Figure 5.3: *Edge Map with Chip ID Removed*

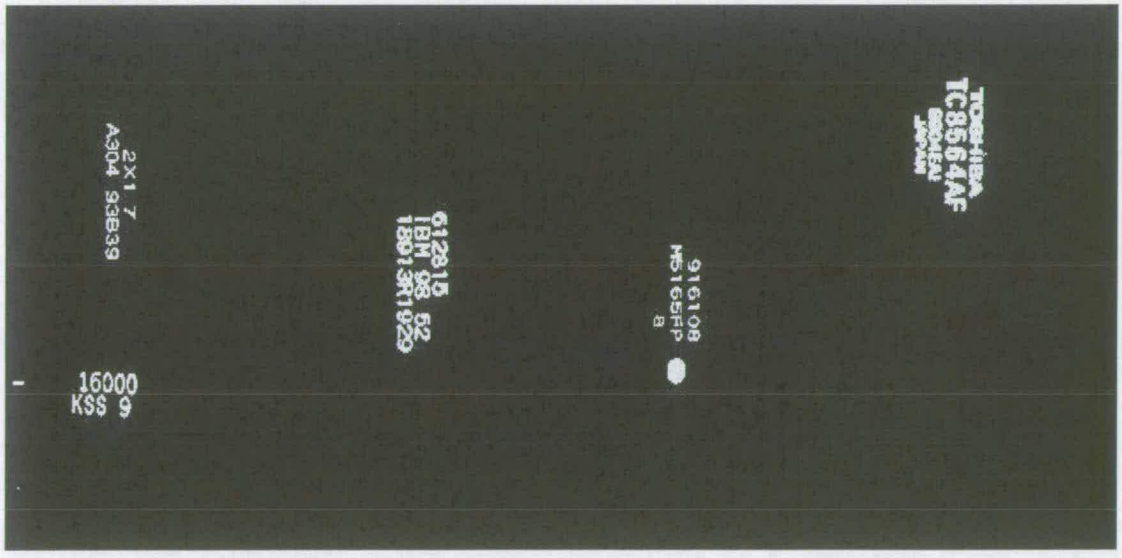

Figure 5.4: *Chip ID Map*

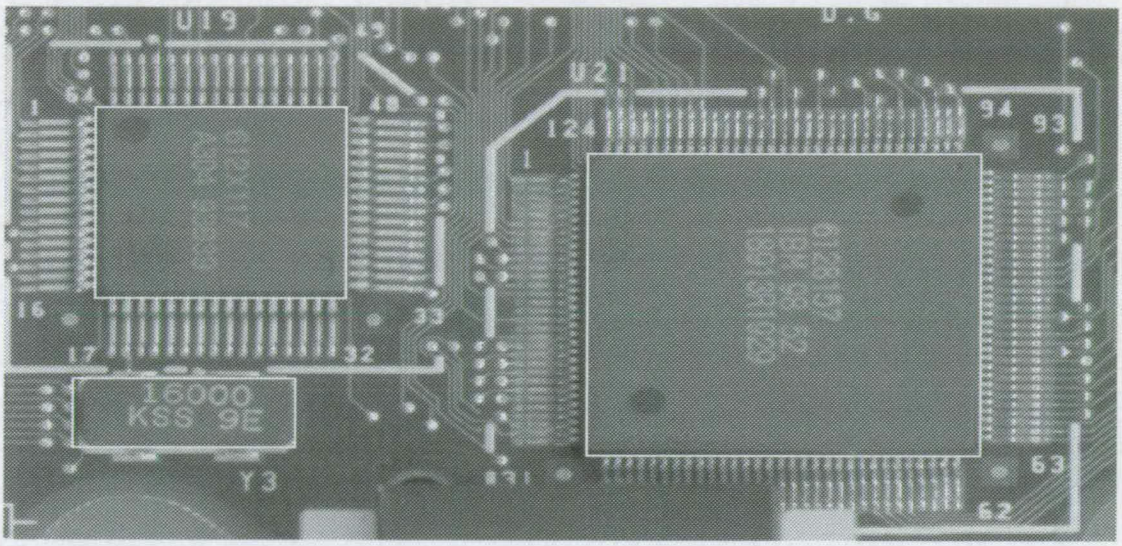

Figure 5.5: *Grey Scale Image with Three ICs Detected Correctly*

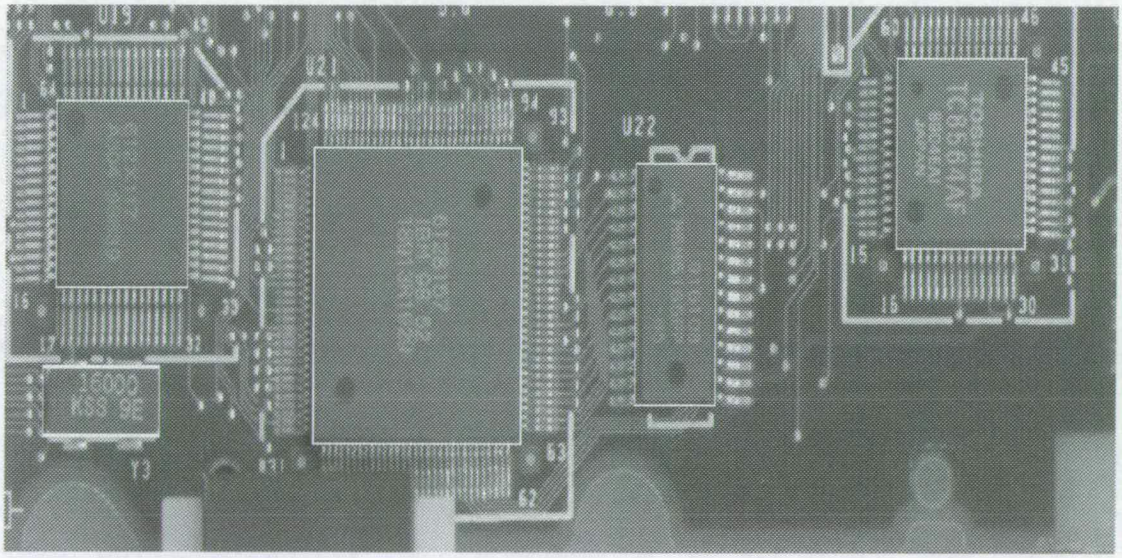

Figure 5.6: *Five IC Grey Scale Image Test Sample with Correct IC Detection*

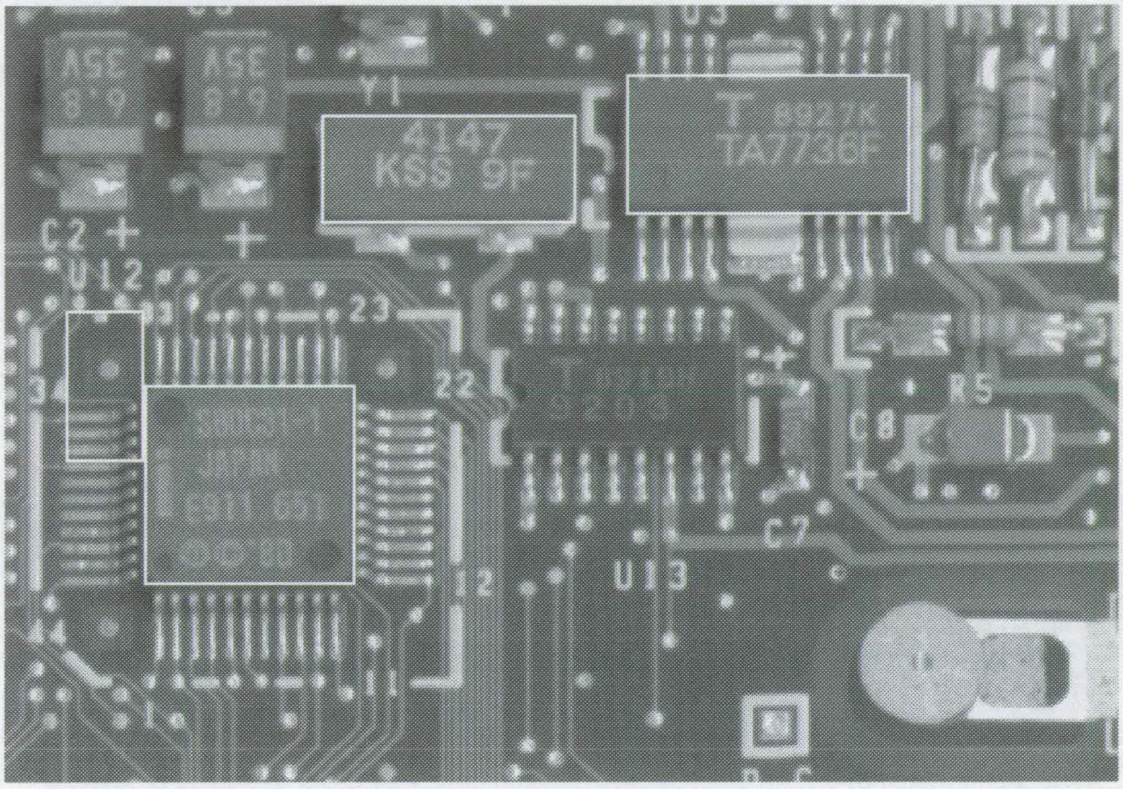

Figure 5.7: *Further IC Detection Result: Test Sample with ICs in Varying Bands of Intensity*

As can be seen from Figure 5.7 the main limitation of the algorithm is that it only detects ICs in a narrow band of intensity levels. IC casings are often of different materials, metallic for instance, and the algorithm presented here is not sophisticated enough to detect ICs within varying intensity bands. Work on a statistical technique for detecting ICs with varying intensity bands is described in section 5.5.

5.3 Implementation on an ARM Platform

The technique has been ported to an ARM7 target SOC platform. The architecture of the target is given in Figure 5.8. Key features of the architecture are the use of a JTAG port for the Multi-ICE debug unit [104] which allows debugging at the source level on the target. Another feature is the use of a high speed Advanced High Performance Bus (AHB) for communication between system elements. The use of a Static Memory Interface allows connection to Flash (32MB) memory for storing executables and connection to SDRAM (64MB) for storing Portable Grey

Map (PGM) images.

The software was initially developed and tested on an Unix Solaris host. The software was then ported to an ARM7 core module target [86] using the ARM Developer Suite (ADS) [105], [106], [107], [108], [109], and [110]. The ARM7 core modules connects directly to the ARM Integrator ASIC Development Motherboard detailed in [85].

The main changes to the code compared to an Unix Sun Solaris implementation are the reading and writing of PGM images to and from SDRAM rather than a file. This process involves the writing the PGM image to SDRAM and then the code reads and interprets the header of the image. A two dimensional array is declared according to the header and then the grey level intensities are read in from SDRAM. The image processing operators such as edge detection and region growing can then be applied and the PGM image is written out to SDRAM with the appropriate header. The block of SDRAM containing the image can then be saved and interpreted by an image processor such as XV [111] or Gimp [112] to check the result. This procedure is given in Figure 5.9 where there is a process for interpreting the image header and reading the image from SDRAM, a process for carrying out the image processing operator, and a process for writing the image header and writing the image to SDRAM.

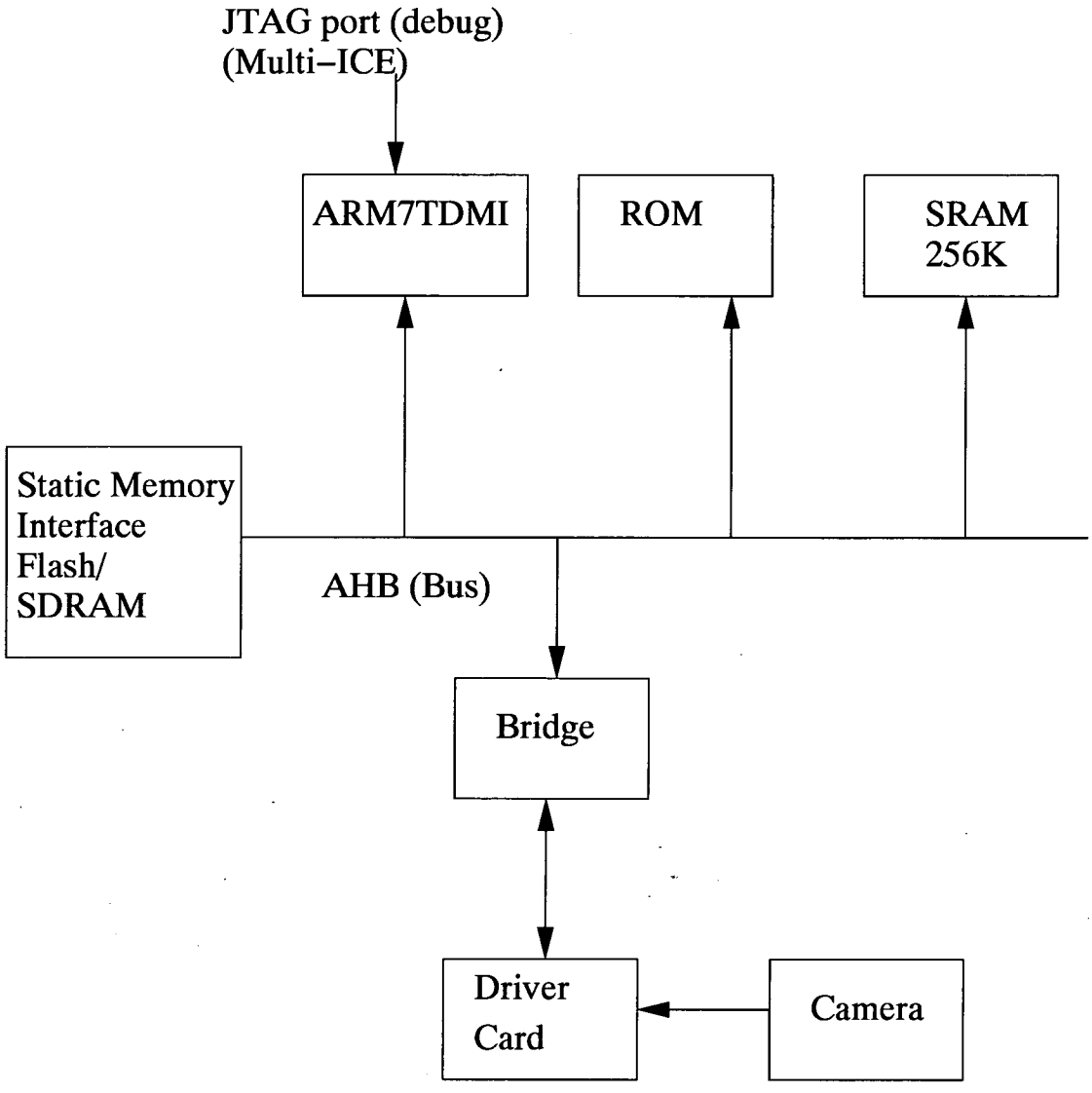

Figure 5.8: *ARM7 Architecture*

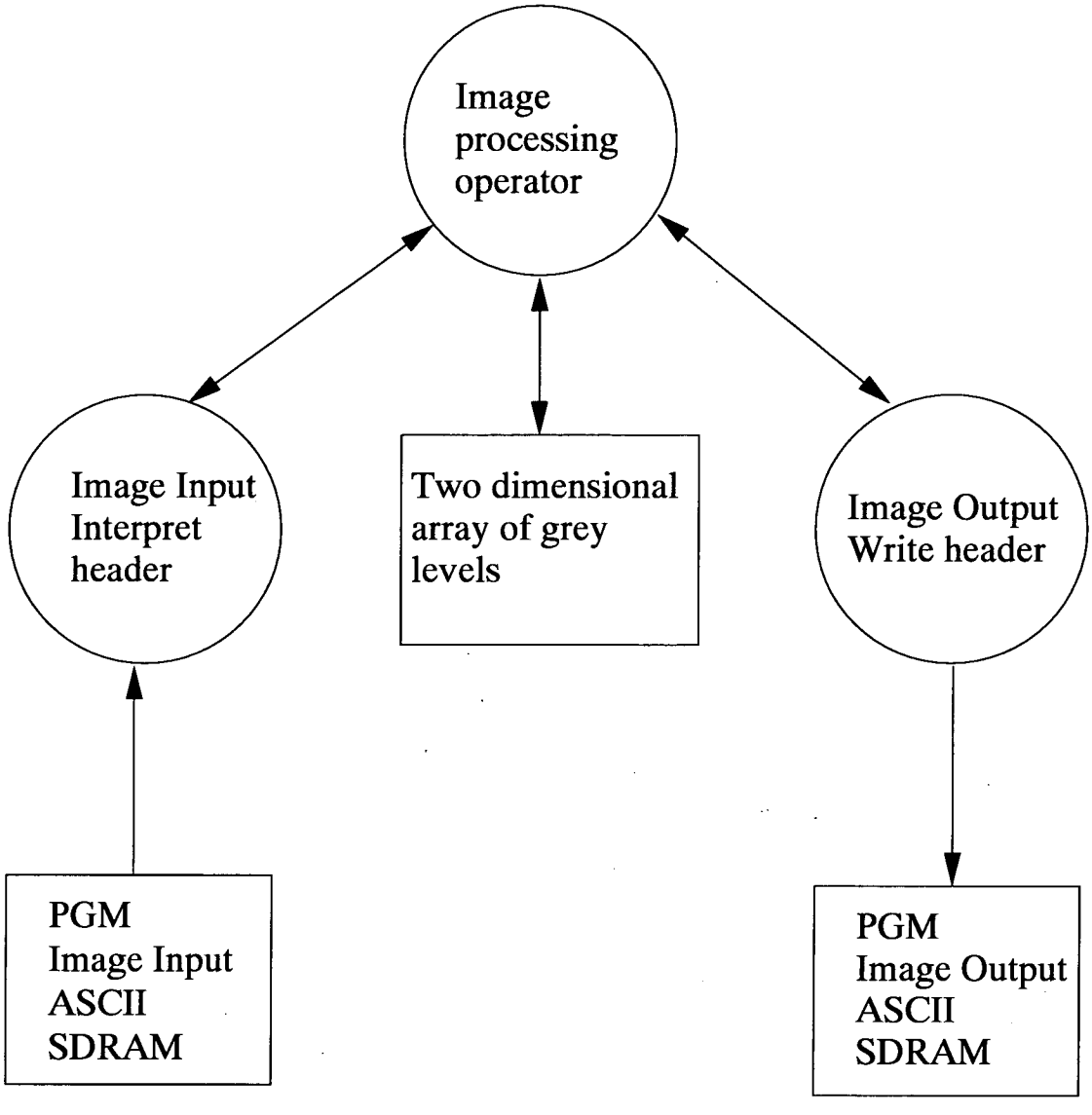

Figure 5.9: *Image Processing Operator on the Target*

5.4 Results

An example PCB image with five ICs of varying size is shown in Figure 5.10. The result of the find IC process on this image is shown in Figure 5.6 and shows that the five ICs have been detected with the borders of the chip area being found.

Figure 5.11 shows the binary map derived from the multi-thresholding of Figure 5.10. Figure

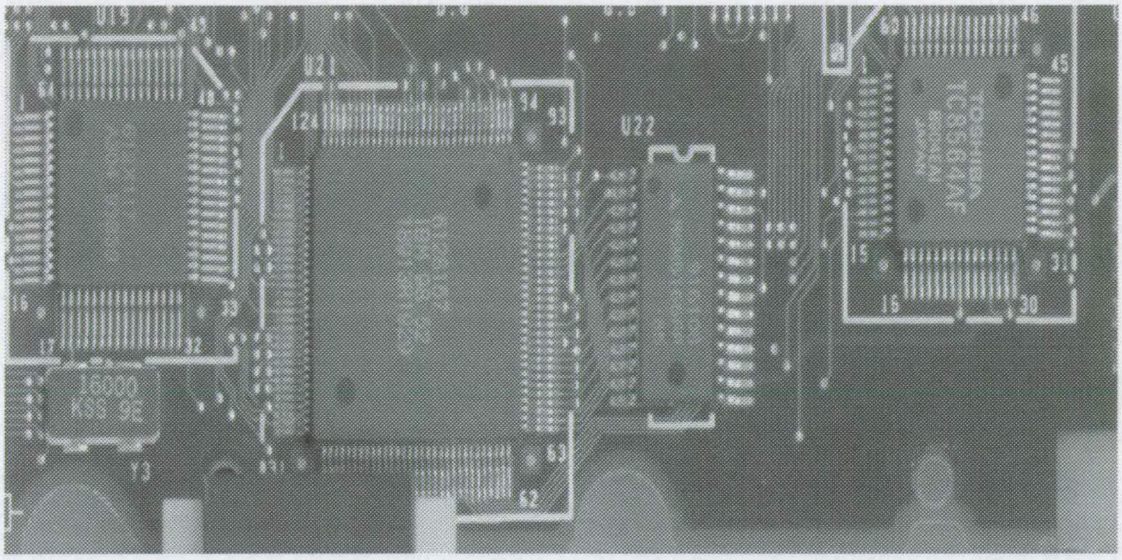

Figure 5.10: *Five IC Image Grey Scale*

5.4 shows the chip ID map derived from the binary map through region growing. This shows that only the chip IDs are present and no other features. This is important because the Chip ID map is used to delete the chip ID from the edge map. This deletion is shown in Figure 5.3 where all the chip IDs have been removed from the edge map. The region growing process can then find the edges of the chips correctly. It should be noted that if chip ID regions were still present then this final region growing step would give inaccurate results.

A further test image result is given in Figure 5.12. This is a much more difficult image to process due to the presence of resistors, capacitors and other non-component features. All three Integrated Circuits have been detected accurately however.

A further test image result is given in Figure 5.7. This image shows the main limitation of the technique: only ICs within a restricted band of grey scale intensities are detected correctly. The IC in the centre of the image is outside the range of intensities of the rest of the ICs and is not detected. The two capacitors in the top left hand corner of the image and the resistors on the right hand side of the image are not detected. This result therefore shows that the technique is robust. However, there is a spurious IC detection box next to the bottom left IC which is due to features in this region registering as a false chip centroid position. The main point is three of the four ICs in the image have been detected correctly to a high degree of accuracy. The AOI machine operator would have to correct any errors which should be a simple process carried

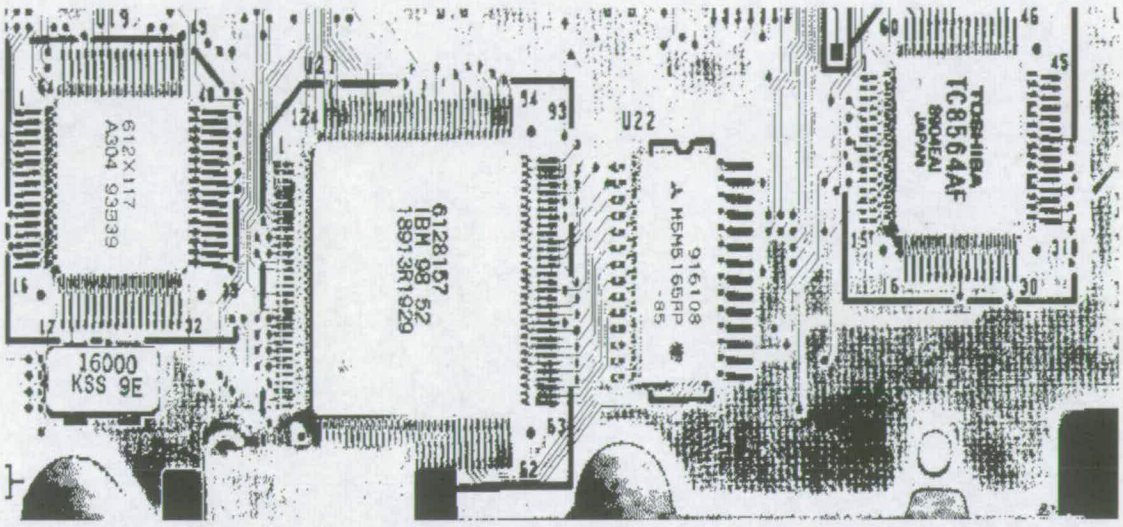

Figure 5.11: *Binary Map*

out through an appropriate graphical user interface.

The SUSAN operator is a key performance critical function in the technique. After porting this operator on an ARM platform the execution time was only 90 seconds for the image shown in Figure 5.5. This is an important result as it shows the execution of the edge detection operator is within reasonable bounds on a target system. When running as a stand alone system without the overhead of the debugger this time will be further reduced.

The number of operations for the edge map image in Figure 5.13 which contains 16410 pixels is 10.4 x 10^9 for the Hough transform. This contrasts with the number of operations for the region growing and find IC processes for the same image which are 1.2 x 10^6 and 1.2 x 10^7 respectively which gives a total of 1.32 x 10^7.

The execution time of the Hough Transform for detecting rectangles is of order O(n^6) where n is the number of pixels in the image. The region growing and find IC processes have greatly reduced complexity and are O(n) and therefore represent a significant improvement in efficiency.

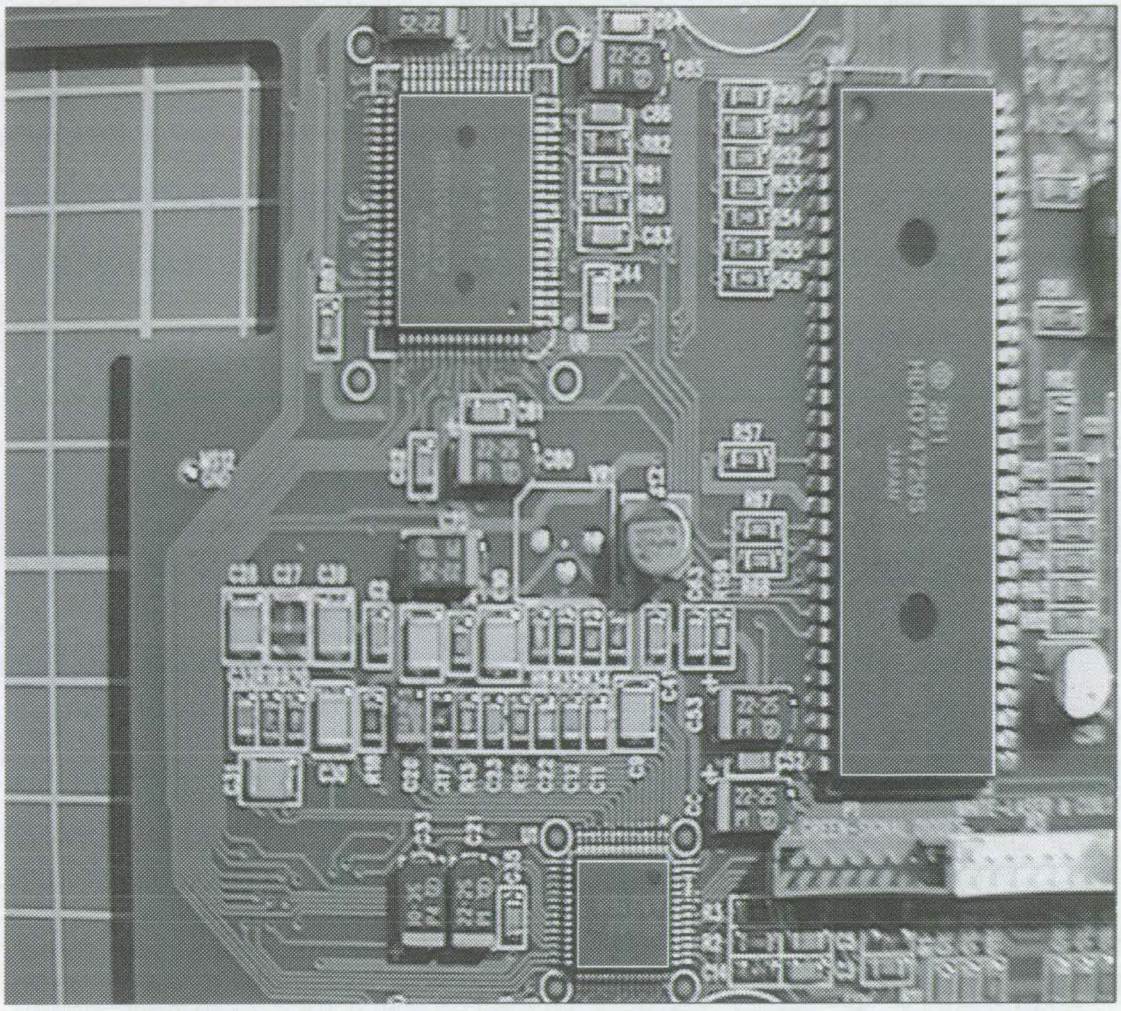

Figure 5.12: *Further Test Sample with Correct IC Detection*

5.5 Enhancements

5.5.1 Further Segmentation Techniques

The image in Figure 5.7 has chips with grey scale values in varying bands of intensity. This result shows that the IC detection process needs to be enhanced to detect all ICs whatever their intensity levels. A solution to this problem is to use the output of the first application of the SUSAN operator to segment the original image. For this problem the edge map is given in Figure 5.14. This process proceeds as follows: if a pixel is present in the edge map then set the pixel in the original image to zero intensity otherwise set the pixel to maximum 255 intensity. This gives a binary image similar to Figure 5.11. The region growing technique is then applied

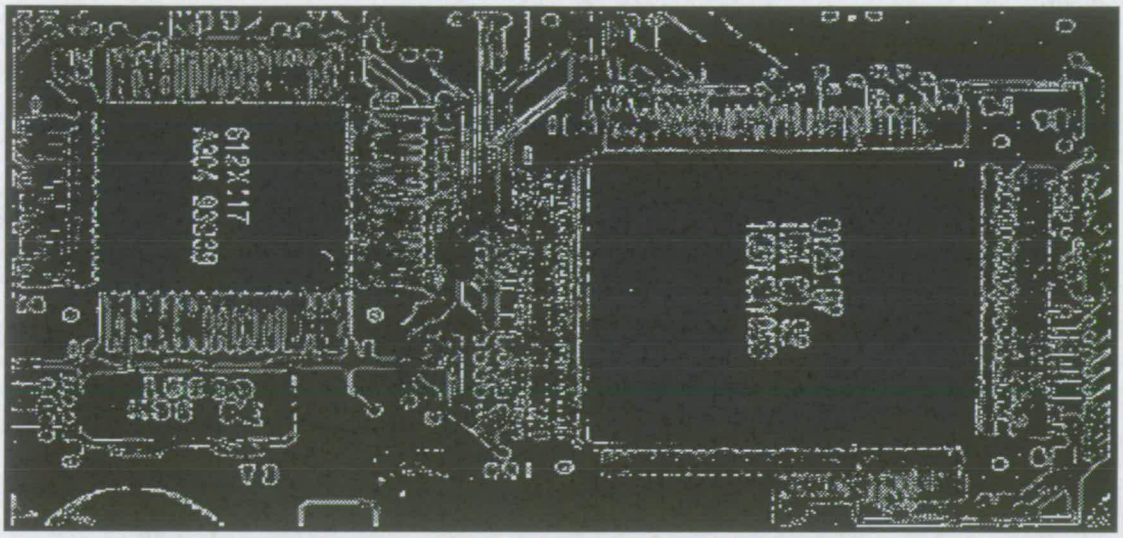

Figure 5.13: *Result of SUSAN Operator giving Edge Map*

as before to this segmented image. The result of this process on the Figure in 5.7 is given in Figure 5.15. This result shows that a further IC has been detected. This process works because the ID of the additional detected chip is present in the edge map used to segment the original image.

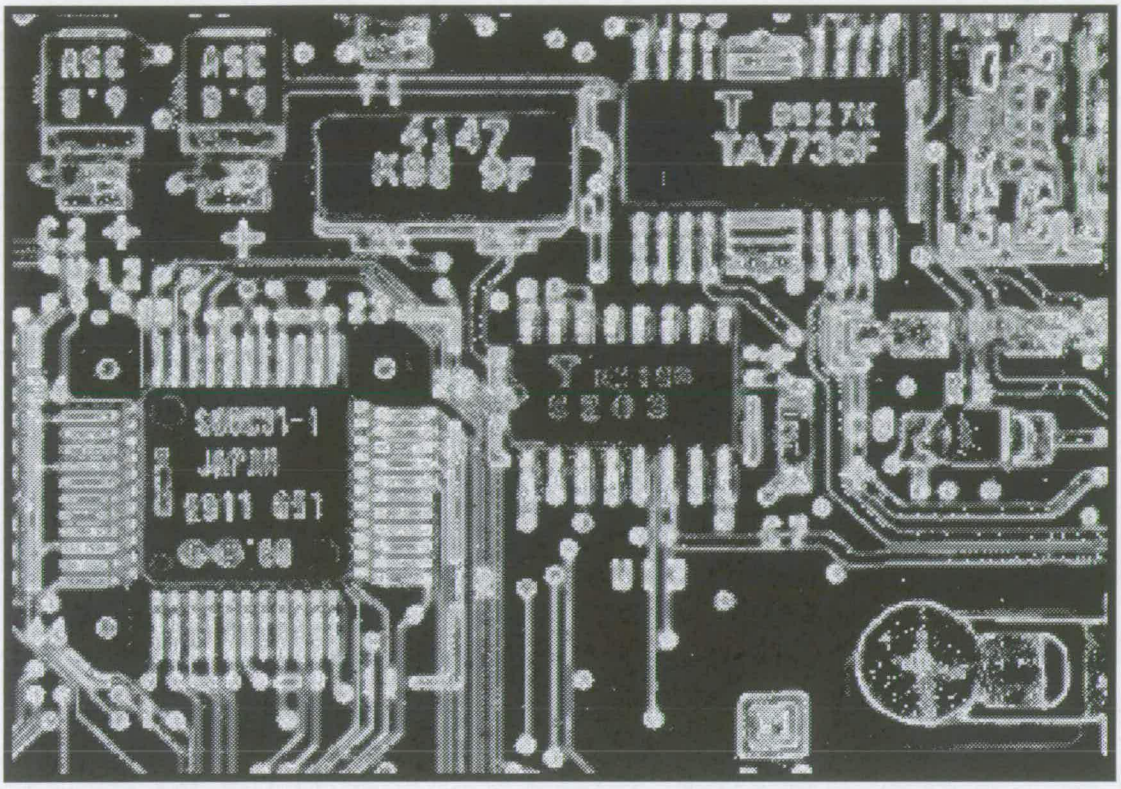

Figure 5.14: *Further IC Detection Edge Map*

5.5.2 Probability Processing

Another solution to the problem of detecting Integrated Circuits in varying bands of intensity is given in [113]. An algorithm for this technique is given in Pseudo code 6. This technique is based on deriving the Cumulative Probability Function (CPF) of an image and then transforming the intensity distribution of the image so that every pixel is directly proportional to its CPF. This process has been applied to multiple test samples such as 5.15 and 5.16 and the results are the same across the test set. Here the process applied to 5.16 is illustrated. The transformed image from the CPF is given in Figure 5.17. This image shows that the Chips have been isolated from the background thus supporting further processing.

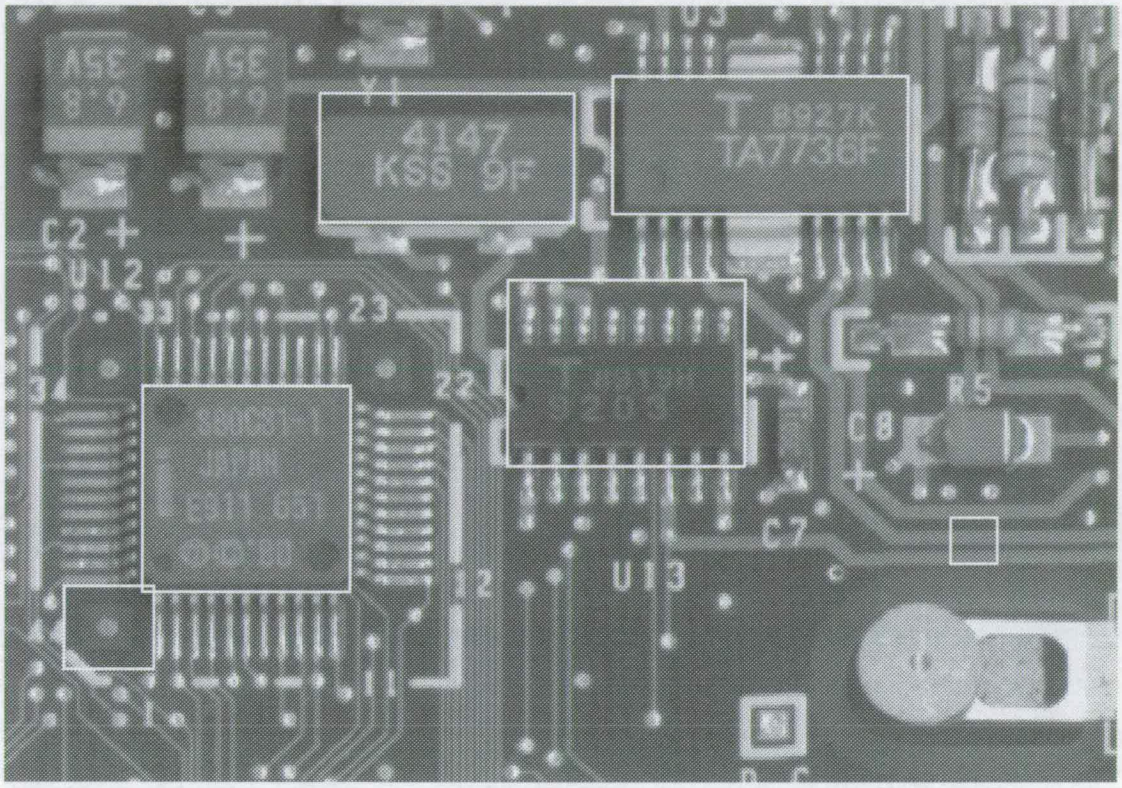

Figure 5.15: *Further IC Detection Result: Detection of ICs in Varying Bands of Intensity*

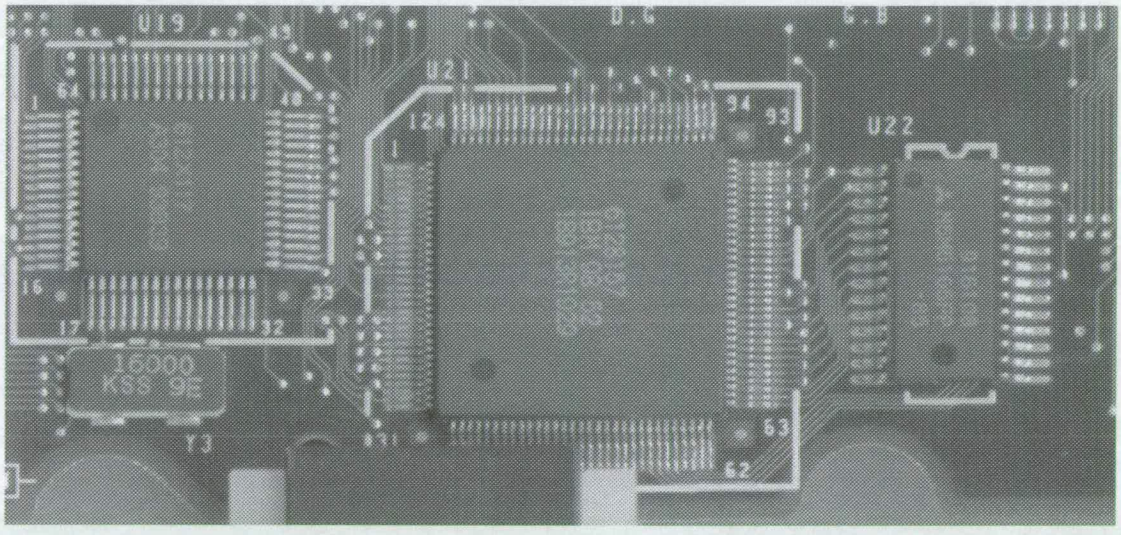

Figure 5.16: *Four IC Grey Scale*

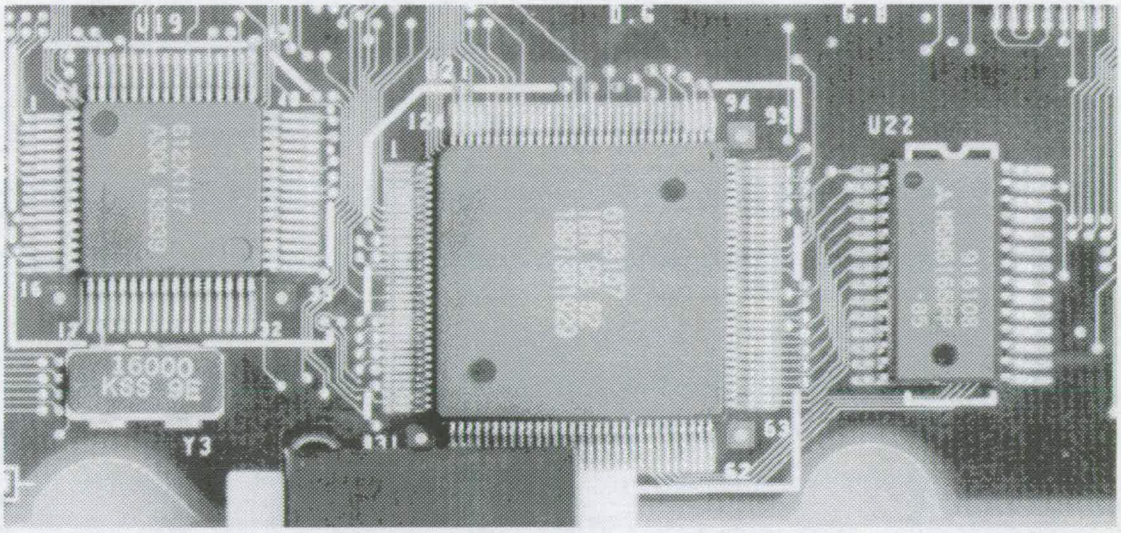

Figure 5.17: *Four IC Transformed*

The graph of the CPF for Figure 5.16 is given in Figure 5.18 and shows the value of CPF increases with the intensity value.

The next stage in the algorithm is to select a Region Of Interest (ROI), a sample of a chip, such as in Figure 5.20. An optimum threshold is then computed from the gradient of the intensity distribution at every pixel. This gradient is available from the sobel magnitude operator. The result of the sobel operator for the test image is given in Figure 5.19. The computation of the threshold is given by Equation 5.1.

$$threshold = \frac{mean(\overrightarrow{sobelimage \cdot ROI})}{mean(sobelimage)} \cdot 1.2 \tag{5.1}$$

This equation shows that each element of the Sobel image and each element of the ROI image are multiplied together. The mean is then computed by totalling the resulting intensities and dividing by the number of pixels. The threshold value is then divided by the mean of the Sobel image and multiplied by a factor of 1.2. This factor has been found through experimentation to give optimal results. Figure 5.20 shows that without the factor of 1.2 the threshold is too low and the chip ID is not isolated and further processing stages in the algorithm will fail to give optimal results.

The ROI is then binarised according to this threshold giving Figure 5.21. This binarised image

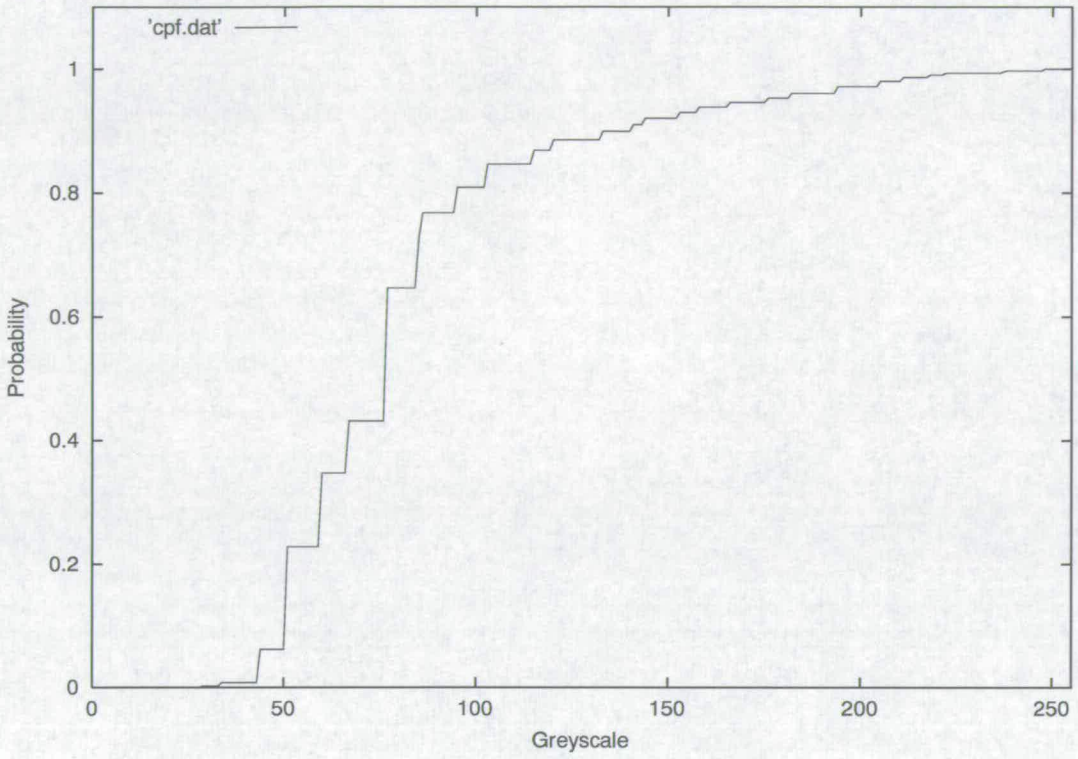

Figure 5.18: *Cumulative Probability Function Distribution for Figure 5.16*

is then inverted to make a mask. The result of this process is given in Figure 5.23.

The next stage in the algorithm is to compute the mean (μ) and standard deviation (σ) [65] of the ROI and the ROI mask. The mean is computed through Equation 5.2.

$$\mu = \frac{\sum_{ij} ROI}{totalpixels(ROImask)} \tag{5.2}$$

In this Equation for every non-zero intensity of the ROI mask a total of the pixel intensities for the ROI is made. The mean is then this intensity total divided by the number of pixels in the ROI mask. The standard deviation is then computed with reference to Equation 5.3.

$$\sigma = \sqrt{\frac{(ROI - \mu)^2}{totalpixels(ROImask)}} \tag{5.3}$$

This Equation uses the computed μ, the ROI intensities and the total number of pixels in the ROI to derive σ. For each ROI intensity μ is subtracted. The total is then squared and then

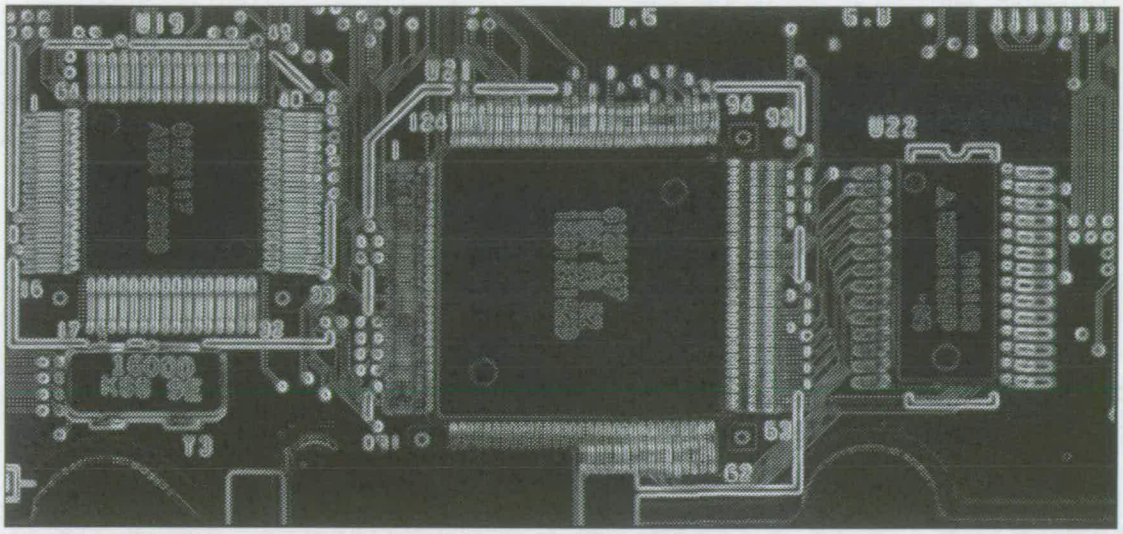

Figure 5.19: *Four IC Sobel*

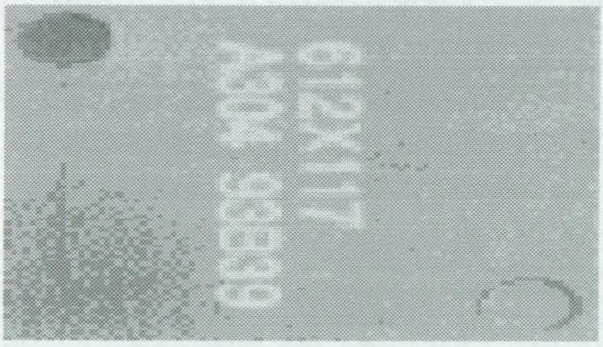

Figure 5.20: *Region of Interest*

divided by the number of pixels in the ROI mask. The square root is then taken of this result to give σ.

Normal or Gaussian distributions can be used for modeling many naturally occuring phenomena [65]. In [113] it is assumed that the intensity distribution of the ROI is approximately Gaussian. A Gaussian distribution can be described using the computed μ and σ of the ROI.

A graph showing the normalised pixel intensity distribution for the ROI, and the Normal or Gaussian probability [65] function from the computed ROI μ and σ as parameters to Equation

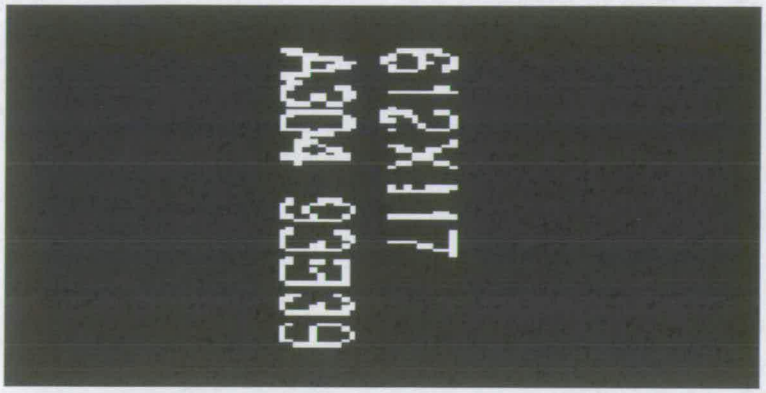

Figure 5.21: *Region of Interest Binary*

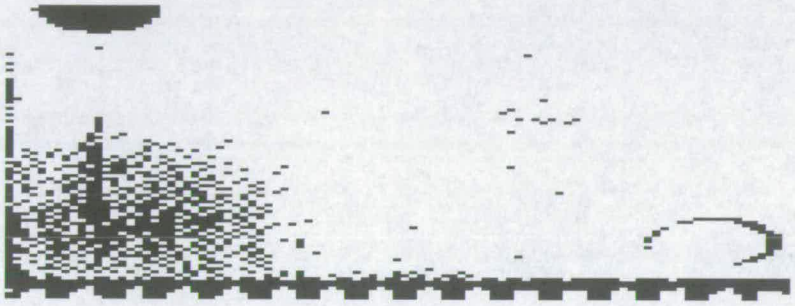

Figure 5.22: *Region of Interest Binary without the Threshold Factor*

5.4, is shown in Figure 5.24.

$$probability(grey, \mu, \sigma) = e^{-\frac{(grey-\mu)^2}{2.\sigma^2}} \tag{5.4}$$

The normalised pixel intensity distribution for the ROI is computed through totalling the number of pixels for each of the 256 possible values. The total for each intensity value is then divided by the maximum intensity to give a distribution with values between zero and 1.0.

The graph in Figure 5.24 shows that the peak of the normalised data corresponds well with the peak of the Gaussian. This probability function is then applied to every pixel in the transformed image. This generates an image whose intensity is directly proportional to the probability that it belongs to the distribution. For the transformed test sample this gives the probability image in Figure 5.25. This image shows that the areas of high intensity correspond well to the locations of the chips.

The probability image is then segmented using Equation 5.5 with the computed σ to give a

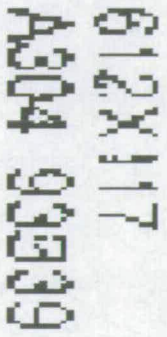

Figure 5.23: *Region of Interest Mask*

threshold.

$$threshold = e^{\frac{-(3.\sigma)^2}{2.\sigma^2}} \tag{5.5}$$

For the probability image this gives Figure 5.26. This final step is performed by selecting a threshold with the difference of grey to mean is equal to three times the standard deviation. This computation of threshold is found through experimentation to give optimum results.

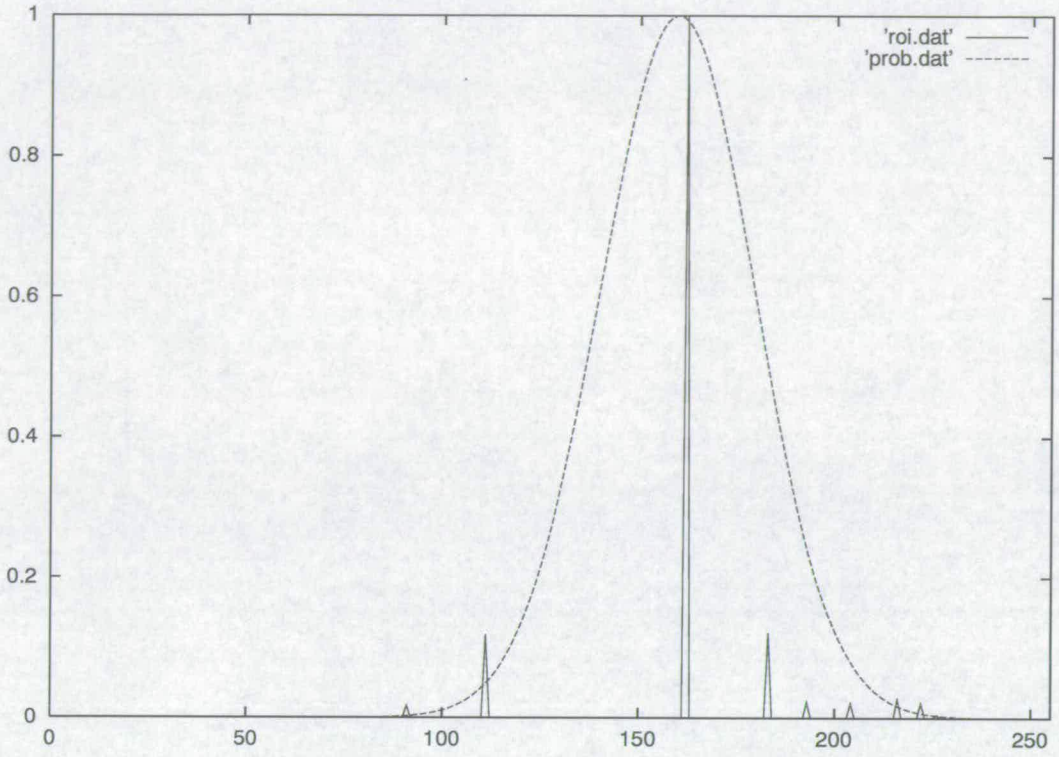

Figure 5.24: *Region of Interest and Probability Function Distributions*

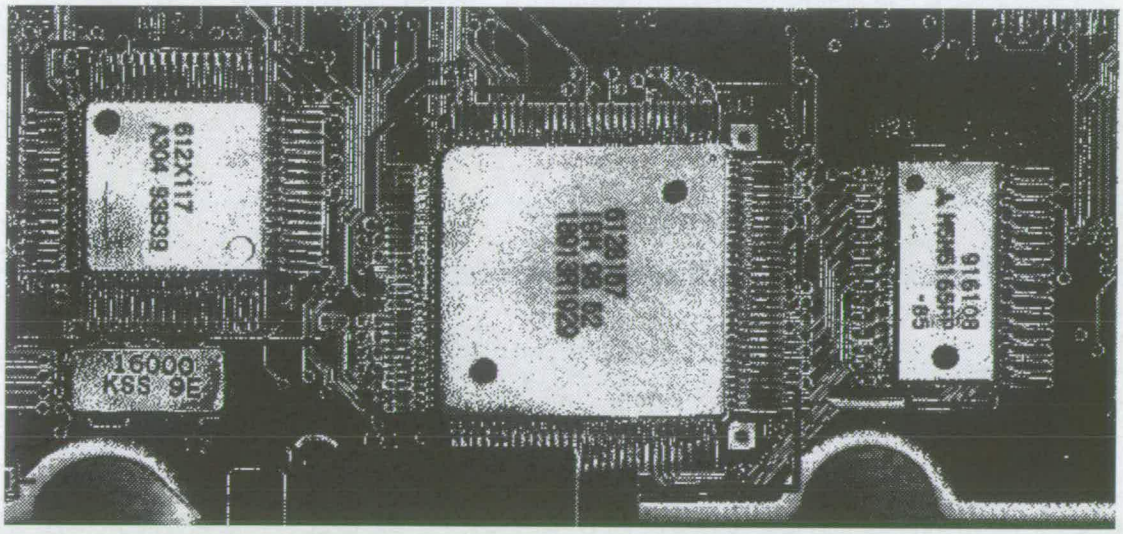

Figure 5.25: *Four IC Probability*

```
Compute total number of pixels in image
Compute histogramme of pixels using number of pixels in image
        loopfor i = 0 to 255
            probabilty[i] = histogramme[i] / total number of pixels
Compute Cumulative Probability Function (CPF) for each grey level
        loop for i = 0 to 255
            CPF[i] = CPF[i - 1] + probability[i]
Transform image according to CPF for each grey level
        loop for i = 0 to image.width
            loopfor j = 0 to image.height
                transformed[i][j] = CPF[image[i][j]] * 255.0
Select Region Of Interest (ROI)
Compute optimum threshold for ROI using Sobel of ROI
        Compute gradient: Sobel operator on ROI
        Compute threshold using equation 5.1 with computed gradient and ROI
Binarize ROI according to optimum threshold
        if ROI pixel > threshold set to 255
        otherwise set to 0
Invert ROI to give ROI mask
        if ROI pixel = 0 then set ROI mask to 255
        else if pixel = 255 then set ROI mask to 0
Compute mean and standard deviation of the ROI using ROI mask
        k = 0
        loop for i = 0 to ROI.width
            loop for j = 0 to ROI.height
                if ROImask[i][j] ≠ 0
                        vector[k] = ROI[i][j]
                        k = k + 1
        ROI.μ = mean(vector)
        ROI.σ = standard deviation(vector)
Apply probability function equation 5.4 to transformed image using computed ROI.μ
and ROI.σ
        loop for i = 0 to transformed.width
            loop for j = 0 to transformed.height
                probabilityimage[i][j] = result of equation 5.4 with parameters transformed[i][j],
                ROI.μ and ROI.σ * 255.0
Compute threshold using equation 5.5 with ROI.σ and multiply by 255.0
Apply threshold to probability image to binarise image
        loop for i = 0 to probability.width
            loop for j = 0 to probability.height
                if (probabilityimage[i][j] > threshold)
                        binaryimage[i][j] = 255
                else
                        binaryimage[i][j] = 0
```

Pseudo code 6: *Algorithm for Probability Processing*

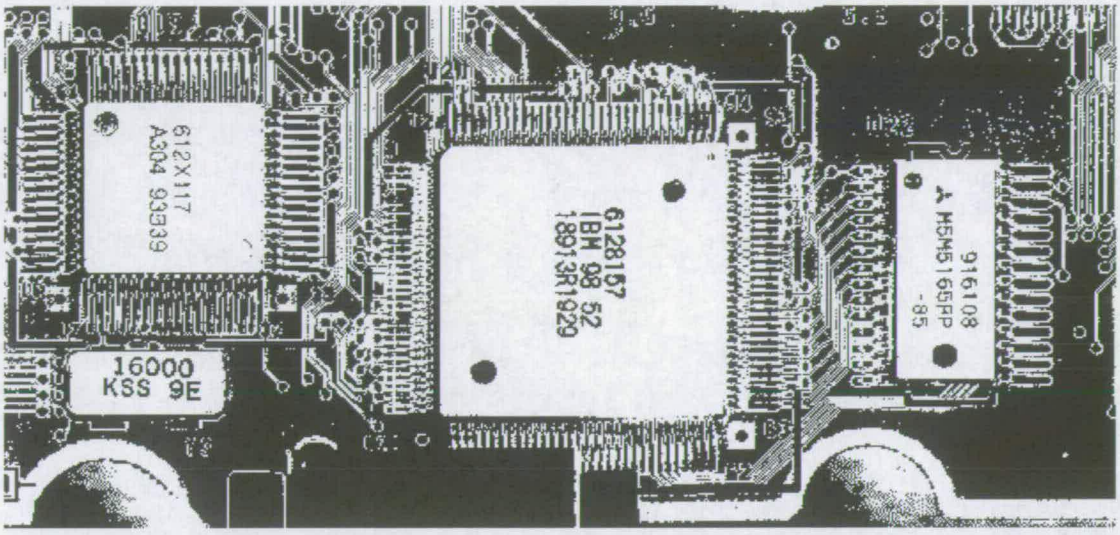

Figure 5.26: *Four IC Segmented*

Once the final step has been performed the binarised probability image can be used as input into the region growing technique to find the locations of the chips.

This work needs to be enhanced to train on multiple chips and this is further work for this thesis. For this enhancement the user will have to select an ROI for each chip with a distinct intensity band. The mean and standard deviation for each ROI will then be computed as before. The Gaussian probability function given in Equation 5.4 will now be a sum of Gaussians, with one term for each chip type. A sample equation will be of the form of Equation 5.6 which is for three chip types.

$$probability(grey, \mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3) = e^{-\frac{(grey-\mu_1)^2}{2.\sigma_1^2}} + e^{-\frac{(grey-\mu_2)^2}{2.\sigma_2^2}} + e^{-\frac{(grey-\mu_3)^2}{2.\sigma_3^2}} \quad (5.6)$$

5.6 Conclusions

This Chapter has presented a novel technique for the detection of ICs within images of PCBs to be employed in an ARM based SOC system for real-time diagnostics. The technique presented which uses region growing has a significant performance improvement over existing techniques such as the Hough Transform. The performance of the original Hough Transform cannot be verified with the resources available for this work due to the technique's computational requirements in terms of memory and processing time.

The key idea of the region growing technique is to use detection of the chip ID on ICs as the basis for performing a region growing search to find the edges of the ICs. Results show that the majority of ICs are detected correctly.

The main limitation of the technique has been shown. This limitation is the fact that only ICs within a restricted band of grey scale intensities are detected. The Chapter goes on to describe two techniques for solving this problem. The first technique solves the problem by using the output of the SUSAN edge detection to segment the original image. This technique works because edges are detected whatever the grey level intensities of the chips. The second technique is a probability processing technique and is much more complex. This technique involves deriving a probability image where the intensity of each pixel is proportional to a Gaussian probability function. This Gaussian probability function is derived from a ROI selected from the original grey scale image.

The segmented image output of these two techniques can be used as input into the region growing process, where the Integrated Circuits in varying bands of intensity will be detected successfully.

The Hough Transform is of order $O(n^6)$ compared to region growing and find IC algorithms which have complexity $O(n)$. This reduction in complexity allows the use of such algorithms in embedded systems such as the one described in this Chapter. The Chapter has presented the porting of the techniques to an ARM7 SOC target. The porting of an object recognition operator to this target system is novel and the feasibility of this approach has been shown.

Chapter 6
The Implementation of an Evolvable Hardware System for Real-Time Image Registration on a System-On-Chip Platform

6.1 Introduction

This Chapter presents an evolvable hardware system for Real-Time Image Registration implemented on a conventional SOC platform. Exploring the use of a Genetic Algorithm for Image Registration for detecting placement errors of components on a PCB was the main goal of the original proposal for this work. This proposal was in the context of a prototype industrial system. The proposal goes further specifying the targeting of the Genetic Algorithm for a System-On-Chip. There are other techniques which can be applied to this type of registration problem and they include Wavelet [114] and Gabor Wavelet [115] transformations. The important point about these transforms is they are rotationally invariant. This makes them suitable for a problem where the rotations of the multiple objects being registered is arbitrary. This can be compared to the Fourier transform [1] processing which becomes computationally complex for this type of problem.

In order to provide flexibility most components of the Genetic Algorithm, which forms the basis of the Evolvable Hardware, are implemented as embedded software and ported to various components on the SOC platform. The chapter describes optimisation techniques in order to achieve real time speed for porting the algorithm on an ARM7 based SOC platform. Results for the execution on the host platform and the SOC target are presented. Through analysis of the results, a modified platform is proposed for the implementation of the Evolvable Hardware system. This enhanced system architecture includes a high performance Digital Signal Processing Intellectual Property Core.

Today SOC devices target high performance applications in which fast time to market is of

prime importance. For this reason SOC platforms are used. On these platforms there is a tradeoff of performance, flexibility and fast time to market. A typical platform usually combines a processor, memory and peripherals around a standard bus architecture such as the AMBA. Specific IP cores could be added to form derivatives targeting specific applications. This procedure reduces design time and increases flexibility.

The correct positioning of Integrated Circuits is a major issue to developers of Automatic Optical Inspection systems, for diagnostics of Printed Circuit Boards. Integrated Circuits will not function correctly unless they are placed on the board with a high degree of accuracy [26]. In this work adaptation to the problem of detecting rotations and displacements of Integrated Circuits occurs through the use of a Genetic Algorithm (GA) for Image Registration. The GA encodes a transform for multiple ICs in a chromosome. The GA then performs a simultaneous search for every IC in the image. This is a registration procedure with a comparison being performed between the real and the reference image. This is processing of a reference golden board and then finding faults on further captured boards. The mechanics of a GA are not described here, see [28] [29] [30] [31] and [32].

Existing work on Automatic Optical Inspection systems for Printed Circuit Board analysis is detailed in [66] and [100]. In [66] breaks in wires and short circuits are detected, and the technique used is referential matching between the stored reference image and the captured test image. In [100] novel structured-lighting inspection technology is used to implement an efficient solution to the detection of solder joint problems.

The system presented here is an Evolvable Hardware system, with adaptation occurring within software using a GA (this is adaptation at the algorithmic level as opposed to circuit level). The adaptive algorithm runs on a Unix Solaris host and is targeted for an embedded ARM7 SOC.

Existing work on Evolvable Hardware concentrates on adaptation at the circuit level. An overview of this work is given in [19]. Evolvable hardware is designed to adapt to changes in task requirements or changes in the environment. The closest application presented in [19] to the work presented here is the data compression chip for electrophotographic printing. In this application a GA is used to search for a set of optimal templates which are used to reconfigure a hardware prediction mechanism.

An example of an adaptive real-time imaging system is [116]. In this system a road traffic sign is detected in real time through the use of a GA. The position and size of the traffic sign are

coded as gene information. It should be noted that the throughput of data in this system is high compared to the system presented here because real-time video is being processed. The application described here processes still images of Printed Circuit Boards. However, the efficiency of the diagnostic software implementation is still a major issue due to the high throughput of samples under test. To analyze this efficiency software profiling of the implementation has been carried out on the host and target. Processor intensive tasks will then be off-loaded to a coprocessor and the real-time constraints of the application can then be met.

This work in this Chapter is presented as follows: in Section 6.2 presents the chromosome encoding for multiple integrated circuits; in Section 6.3 and 6.4 present the GA and optimization stages; in Section 6.5 presents execution time statistics for the implementation on the host; in Section 6.6 presents execution time statistics for the target implementation; in Section 6.7 presents an enhanced architecture for the system; and Section 6.9 draws some conclusions from the work.

6.2 Multiple IC Chromosome Encoding

The algorithm presented here processes chromosomes which represent multiple Integrated Circuits and their rotations and displacements. An example Multiple IC encoding chromosome for three ICs is given in Figure 6.1. This 33 bit chromosome encodes a transform for three IC's. For each IC transform 11 bits are used with the first three bits encoding a rotation with two bits for the rotation and a single bit for the sign. This is a sign and magnitude encoding. Possible rotations are therefore in the range +/- 3 degrees with respect to the chip centroid position.

The next four bits encode an offset in the X axis with three bits for the value of the offset and a single bit indicating the sign. The following four bits encode an offset in the Y axis in the same format as for the X axis. Possible displacements are therefore in the range +/- 7 pixels around the chip centroid position. Similar to the rotation encoding this is a sign and magnitude encoding.

Given the test set-up specified in section 1.5 this range of rotations and displacements, in terms of actual chip placement dimensions, correspond to the placing of a chip within 1mm of accuracy. This accuracy is with reference to correct chip centroid placement position. This degree of accuracy is essential because of the fine width of the chip legs on surface mount components. Further, these rotations and displacements were used to meet the accurate placement

requirements specified by the industrial partners for this AOI development.

In Figure 6.1 the first IC has rotation 3 degrees and a displacement of -3 pixels along the X axis and -3 pixels along the Y axis with reference to the IC centroid position.

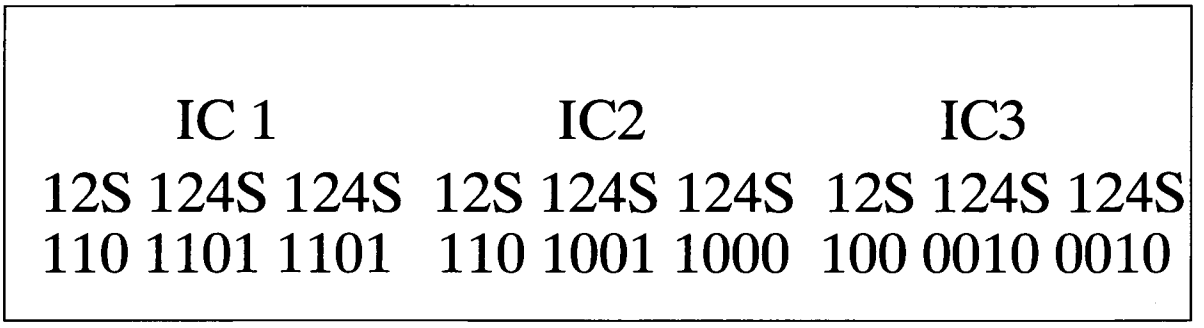

Figure 6.1: *Multiple IC Chromosome Encoding*

6.3 Algorithm Implementation

This section presents the main features of the GA. Pseudo code for the algorithm is given in Pseudo code 7 and Pseudo code 8 which also shows software task partitioning which is discussed in Sections 6.6 and 6.7. A flow diagram for the process is shown in Figure 6.2. The algorithm is also described in detail in [26].

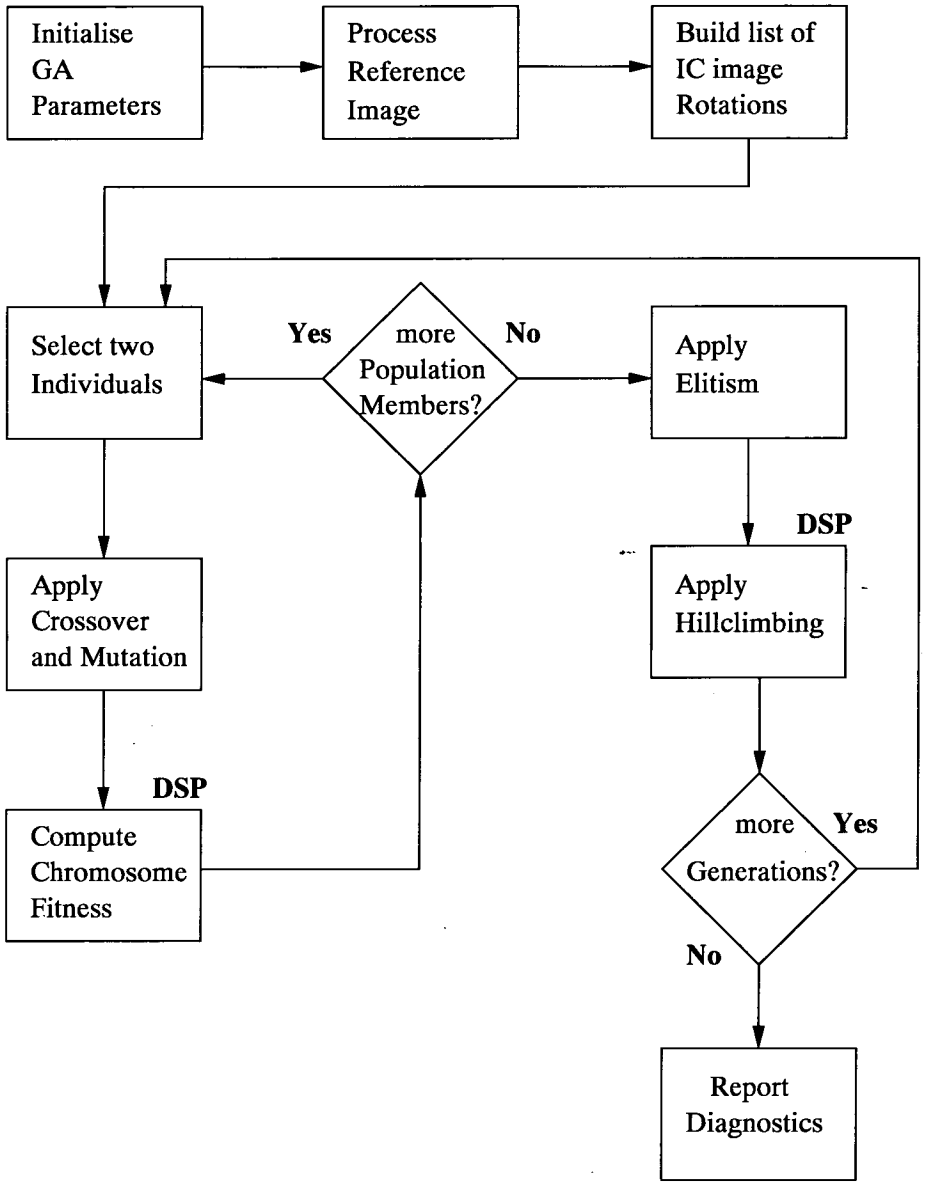

Figure 6.2: *Flow diagram for the Genetic Algorithm for Image Registration*

```
START BLOCK A /* Initialise GA Parameters*/
        Number of generation for run
        Maximum allowable chip rotation angle
        Crossover probability
        Mutation rate
        Population size
END BLOCK A

START BLOCK B /* build a list of reference images of Integrated Circuits */
                Process reference image
                  Build a list of reference ICs using IC detection information
END BLOCK B

START BLOCK C /* pre-compute rotations */
        Loop for 1 to number of ICs
                take an IC from the reference image list
                Loop increment for 1 to max rotation
                    rotate IC image by increment degrees
                    store IC image in rotation linked list
                End loop
        End loop
END BLOCK C
```

Pseudo code 7: *Multiple IC Chromosome Encoding Algorithm Part 1*

The algorithm begins with the initialisation of parameters. The next section involves the building of a list of IC images and then performing the pre-computation rotations on them. In the registration of multiple ICs there is a small number of rotation angles, so incremental image rotations are stored in a list for later reference in the fitness function. Image rotations are not therefore performed in the fitness function of the GA as it has all been done in the pre-computation step. This is a key feature of the algorithm.

The main body of the GA then follows with operators such as selection, crossover, mutation and the fitness function. The selection method used in the implementation is roulette wheel selection. The crossover method used is single point crossover which gives partial solution combination. Single bit mutation is used to ensure against the derivation of uniform populations. The fitness function then involves a match procedure between captured and reference IC image. This procedure involves matching each pixel grey scale level in the captured image with each pixel grey scale level in the reference image. The fitness is then the number of pixels which have matched divided by the number of pixels in the reference image. This process is

performed with reference to the IC image pre-computation list.

The main part of the algorithm then completes with the use of elitism and hill climbing operators. Elitism ensures the survival of the fittest member of the population to successive generations. The hill climbing operator is an extra local search with chromosomes modified in small stages to achieve faster convergence.

After the main loop the final stage is a report to the operator stating the Integrated Circuits for which a match has not been found in the search procedure. If a match has not been found within a set number of generations it is assumed that the Integrated Circuit positioning is outside of tolerance.

A sample image processed by the GA is given in Figure 6.3 which is an edge map of a single IC. This image is the result of the second application of the SUSAN operator to a grey scale image. Edge maps of this type are processed by the image registration procedure because of the reduced number of pixels to process, when compared to a grey scale image or the result of a single application of the SUSAN operator. The edge map used highlights important features such as the borders of the chips and the chip ID.

6.4 Algorithm and Optimisation Stages

The computational complexity of image processing operators and their associated execution times, within the framework of a GA, is a major issue. In this work reduction in execution time has been achieved in four main stages:-

i) Reduce number of pixel transformations in the fitness function.

For typical Image Registration problems the number of pixel transformations is large. For this problem the number of pixel transformations can be reduced through searching in a local area of each IC, and therefore the whole image is not processed. This local processing is achieved through use of chip position information obtained through IC detection work which is described in [26]. Through having the chip position of the chips it is possible to perform transformations in an area local to the IC. This localised processing gives a major optimisation of the algorithm.

ii) Restriction of the search space.

The search space for the GA is reduced using knowledge of the application area. The rotations

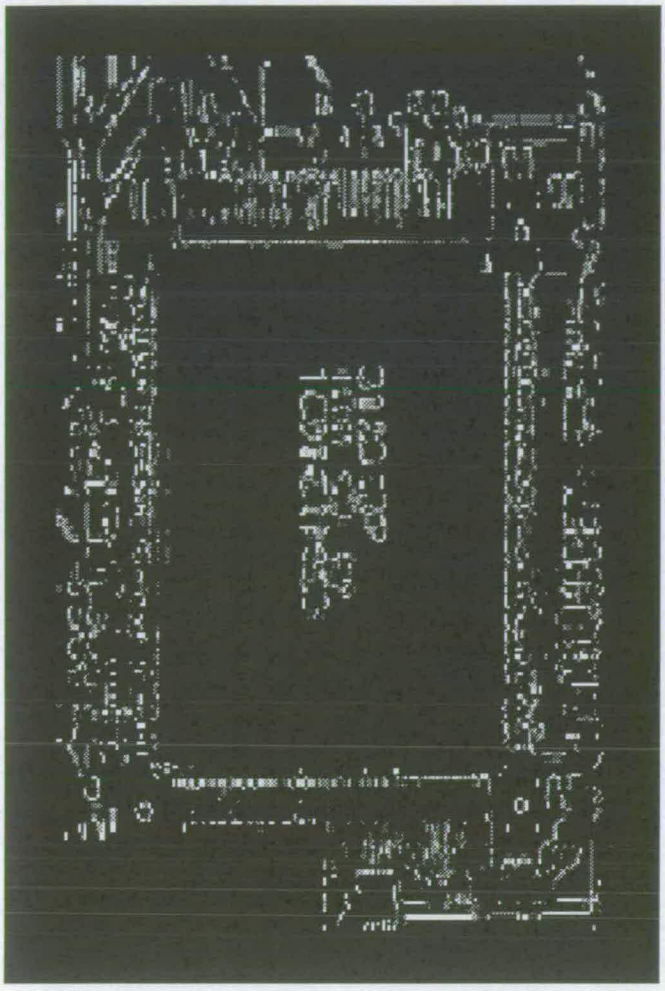

Figure 6.3: *Edge Map of a single IC*

and displacements of ICs have to be within very fine tolerances for the component to function correctly. Therefore the size of the chromosomes can be restricted and thus the search space is reduced.

iii) Pre-computation of IC rotations.

A significant amount of time could be spent rotating images of ICs. In the registration of multiple ICs there is a small number of rotation angles. So an effective technique for optimisation of the fitness function is to pre-compute IC rotations and store the rotated IC images in a list for later matching.

The fitness function then involves a match procedure between captured and reference IC image. This is a key part of the algorithm which significantly reduces computation time for each

generation of the GA. This process is performed using the pre-computed IC image list which stores the rotated reference IC images for matching to the captured IC images. When the chromosomes are decoded, the value for the rotation is used to index into the reference rotated pre-computation list. The captured image is then offset according to the chromosome displacement values. The captured image and the pre-computed reference image are then matched. This matching procedure involves checking the pixel intensity level of each pixel between the transformed captured image and the pre-computed reference IC image. The fitness is then the number of matched pixels divided by the number of pixels in the reference list image. IC image rotations are not therefore repeatedly computed as the algorithm runs.

iv) Use of elitism and the hill climbing operator.

Elitism ensures the survival of the fittest member of the population to successive generations.

The hill climbing operator modifies transforms in small stages to achieve faster convergence. This operator assists in the evolution of chromosomes by modifying them by small amounts. This modification corresponds to moving up or down the solution landscape in small stages. The evolution of the chromosomes was observed and it was found that very close to optimal solutions were being derived. Through small local modifications to the chromosomes convergence can be achieved in much less time than only using a standard GA. The values of rotation and displacement are modified by a single degree or pixel offset. If the fitness of the chromosome is increased then the fitter chromosome is re-encoded and inserted back into the population.

6.5 Execution Time on the Host System

The GA was run for a multiple IC chromosome encoding for six ICs with the following parameters:-

chromosome length: 66 bits
population size: 50
crossover probability: 0.6
mutation probability: 0.05

Our investigations have shown that these parameter values give optimal results. The population

size and crossover probability are standard parameter values for a GA as defined in [32]. The standard mutation rate is 0.001. However, for this registration problem the computation gets stuck in a local minima with this low mutation rate, so a much higher value of 0.05 is used. This higher mutation rate gives faster convergence of the algorithm.

Average time to check orientation and displacement of all six ICs with a test sample of 5 runs was 534 seconds with convergence achieved in an average of 21 generations. The evolution of best and worst runs are given in Figure 6.4. For the best run convergence is achieved in 14 generations, and 356 seconds of CPU time. The worst run took 28 generations to converge, and 884 seconds of CPU time.

With a low mutation rate of 0.001 a typical run to convergence takes 42 generations. Further test results show no convergence with this low mutation rate after over a 120 generations. This contrast sharply with the higher mutation rate where the algorithm converges consistenly in less than thirty generations.

The graph shows chromosome fitness of the fittest chromosomes (labelled 'best') and the fitness for the worst run (labelled 'worst') of all 50 chromosomes. The fitness of the chromosomes stays constant for a period and then increases in stages. This corresponds to a transformation being found for each of the six Integrated Circuits as execution proceeds.

The results are in reasonable bounds for a real-time application where processed boards will usually have upwards of 20 ICs. Further work for this thesis is to expand the test sample for this number of ICs.

6.6 Execution Time on the Target System

The ARM7 is rated at 36 MIPS [80] but does not support floating point operations in hardware [117]. The rotation code from the host uses floating point arithmetic. Therefore to reduce execution time the rotation library code was recoded, for integer operations only, to give a further execution time optimisation.

Psuedo code for the rotation code is given in Pseudo code 9. In this code i and j are integers and are multiplied by 2^{16} to give an integer representation of the sin and cosine of the rotation angle. The following loop is complicated but the main point is that the calculations of the new pixel coordinates involve integer operations and shifts rather than floating point operations. The

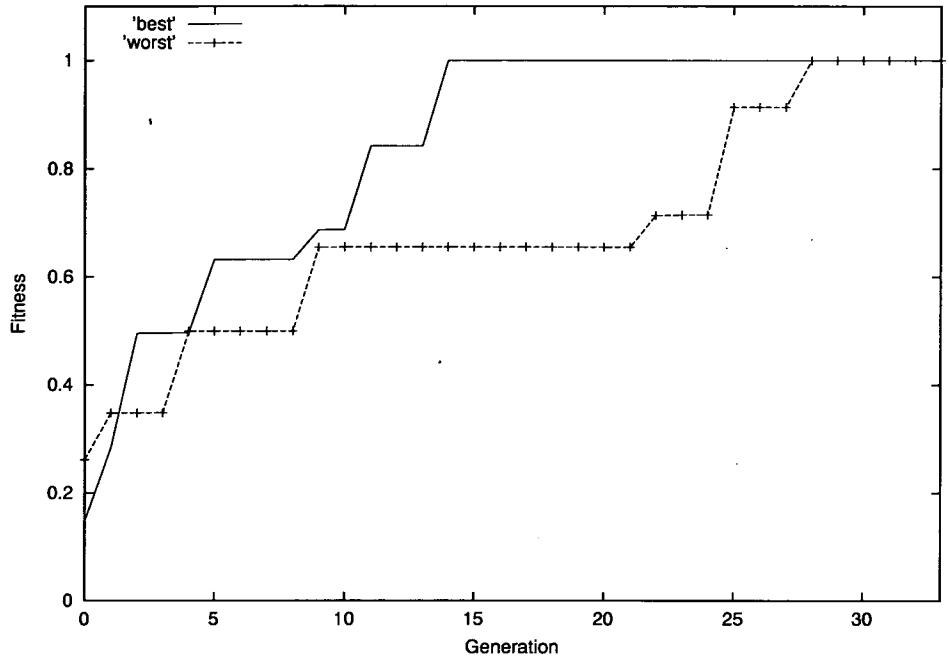

Figure 6.4: *Convergence for a multi-ic chromosome encoding*

calculations of nNewX and nNewY involve a right shift by 16 positions because of the original multiplication factor for i and j.

The code was executed on an ARM7 SOC target with the same parameters as the test on the host system. With a population of fifty individuals the GA takes on average 1400.88 seconds for a generation. Software profiling was then carried out to find which parts of the code were taking significant portions of the total execution time. It was found that the fitness function and the hillclimbing function were taking almost the entire CPU time to compute. Through further profiling the crossover, mutation and selection operators were found to be taking negligible amounts of time. These results are illustrated in Table 6.1 , and shows results for 5 generations. The table shows the cumulative total amount of CPU time for the generation, and the cumulative amount of time spent processing the fitness and hill climbing operators. The percentage of total CPU time spent matching pixels for fitness computation is also given. Note: the amount of time to process a generation varies, due to the varying number of hill climbing operations performed.

Results show that the hill climbing function is taking a greater proportion of the CPU time. In theory a good technique is therefore to run the GA for 10 generations to obtain an approximate solution, and then run the hill climbing operator to find exact solutions. However, host experi-

Gen	*Total CPU Time(s)*	*Total Fitness(s)*	*Total Hill Climbing(s)*	*Total Pixel Match(%)*
1	1183.87	637.81	539.86	83.78
2	2595.45	1272.55	1302.72	83.48
3	3757.13	1910.44	1812.15	83.41
4	5037.27	2547.04	2441.21	83.38
5	7004.38	3204.86	3734.59	82.87

Table 6.1: *GA on ARM: Execution time analysis*

ments show that this approach does not produce any reduction in CPU time for the algorithm. The algorithm takes a greater number of generations to converge in this case, specific results are runs taking over 40 generations compared with a worst case run of 28 generations on the host system. The results also show that a significant proportion of the the fitness and hill climbing functions is spent matching pixels between the reference and captured images (about 83 % of total CPU time).

From these results it is clear that to meet the performance targets of the application the processing time of the fitness and the hill climbing functions will have to be significantly reduced. Therefore in this work we propose the use of a high performance Digital Signal Processor IP core, which is based on the Texas TMS320C6000 series, to speed the execution of the fitness and hill climbing function. This enhanced architecture is given in the next section.

The speedup necessary from the DSP implementation, to give the required performance, can be calculated by comparing the performance of the host implementation, with the target execution statistics. The host implementation results given in Section 6.5 are on average 21 generations, and 534 seconds of CPU time to achieve convergence. This result is approximately 25 seconds a generation. Therefore the DSP will have to be used which speeds the execution of the fitness and hill climbing operators by a factor of approximately 56 times (the target generation time divided by the host generation time: 1400 / 25 = 56).

6.7 The Enhanced Target Architecture

The enhanced target architecture of the proposed system is given in Figure 6.5. The architecture shows the use of the Digital Signal Processor IP core for running the tasks identified in the previous section. The tasks are blocks marked as E and G on the pseudo code in Pseudo code 7 and Pseudo code 8. All other tasks are indicated as being computed on the ARM7. However,

the image rotation pre-computations could also be off-loaded to the DSP to reduce the amount of time the code takes to setup before the main processing begins.

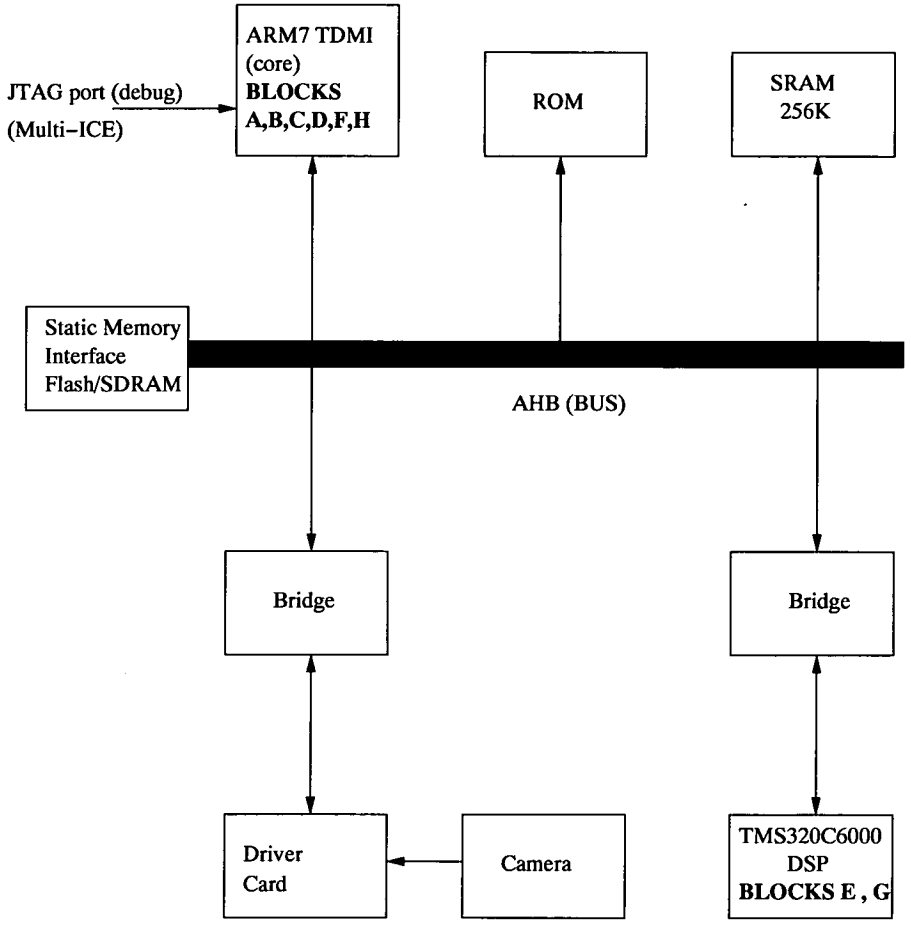

Figure 6.5: *Enhanced Architecture*

The architecture of the DSP IP core is given in Figure 6.6 and is also described in [81]. The architecture is a high performance Very Long Instruction Word (VLIW) architecture. In this type of architecture multiple, independent functional units are used and multiple instructions are packaged into one very long instruction. For example a VLIW instruction may include two integer operations, two floating-point operations, two memory references and a branch, see [61] for further details.

This architecture consists of three main parts: the CPU, peripherals, and memory. Functional units operate in parallel in the Data Paths. The units communicate using a cross path between two register files, each of which contains 16 32-bit registers. Program parallelism is defined at compile time because there is no data dependency checking done in hardware during run time.

The 256-bit-wide program memory fetches eight 32-bit instructions every single cycle. The devices come with on-chip program and data memory, which may be configured as a cache. Peripherals include a Direct Memory Access (DMA) controller.

To show that the use of a DSP IP core will give the required enhanced performance the code for the fitness function pixel match procedure was analysed. This analysis involved converting the ARM assembly language for this function, see [105], into Texas assembly language, described in [82] and [83], and then computing the total number of clock cycles taken by the code. The number of cycles taken by the pixel match code was found to be $79.9 * 10^9$.

The previous section has shown that 82 per cent of the execution time is spent in the pixel match procedure, which for five generations on the target gives a total of 5804.44 seconds of CPU time. The high performance TMS320C6416 is rated at 600MHz giving a 1.67ns instruction cycle time. The following calculation gives the speedup factor if the pixel match code is run on the DSP:-

speedup factor = function total time / (total cycles * cycle time)

speedup factor = $5804.44 / (79.9 * 10^9) * (1.67 * 10^{-9})$

= 43 times

In Section 6.6 we showed that the speedup required was 56 times. However, the assembly code transformation from ARM to Texas assembler is suboptimal. The assembler will apply many complex optimisations which will give the speedup factor required. Full details of these stages are given in [82].

It should be noted that for the final prototype system the software will be enhanced with a user interface function which will run on the ARM7. A greater portion of the processing will therefore be carried out on the ARM7 than is presented here. A high bandwidth connection between the DSP processor and the ARM system will be required due to the volume of image data being processed. A JTAG port for the Multi-ICE debug unit is used which allows debug at the source level on the target. Another feature is the use of a high speed ARM Advanced High Performance Bus (AHB) for communication between system elements. The use of a Static Memory Interface allows connection to Flash (32MB) memory for storing executables and connection to SDRAM (64MB) for storing Portable Grey Map (PGM) images.

6.8 Partitioning of the Genetic algorithm

The whole of the image registration system code was converted to run on the Texas EVM6201 DSP platform. In this section the speed up factor for this cross platform conversion is shown. A description of a partitioned system is then given.

Due to memory restictions on the DSP board the registration process could only be performed on two ICs. This is due to the image rotation pre-computations taking up memory. To save memory the image library's image memory allocation functions were changed to make the representation of a single pixel a character rather than an integer (from four bytes per pixel to one).

Further changes were made to the memory allocation for the representation of the GA's chromosomes from the use of C's *malloc* function to the *calloc* function. The *malloc* function allocates space for a single object while *calloc* allocates space for an array of objects. If *malloc* is used memory allocation problems can occur where a pad is left in memory between elements of a structure. This can lead to corrupti on of a structure element when the last member of the previous structure in memory is written.

Table 6.2 gives the results for the image registration implementation on the DSP for one and two ICs. The values is seconds give the amount of time for a generation. The average amount of time for a generation for a single IC is 111s and for two ICs 170.4s. Therefore for a six IC test the average amount of time per generation can be computed as 111 + (5 * 59) = 406s on average per generation. The speedup factor when going from the ARM platform to the DSP platform is therefore 1400/406 = 3.5. Therefore, the EVM6201 DSP platform does not meet the speedup factor required as specified in section 6.7. So, for the final system a more powerful DSP will have to be used such as the TMS320C6416.

gen	*1 IC*	*2 IC*
0	108s	170s
1	95s	260s
2	110s	150s
3	107s	125s
4	135s	147s

Table 6.2: *GA on DSP results*

Having shown the speed up factor when converting between architectures the software was par-

titioned across the ARM platform and the DSP platform. Communication between plaftforms occurs through the file system on the PC. The fitness function is performed on the DSP while the rest of the genetic algorithm is performed on the ARM. The DSP stores the images of the ICs and receives rotation and displacement parameters for each GA population member from the ARM. The DSP then computes the fitness measure and passes back the fitness for a chromosome. The GA can then proceed as before deriving transformations with the fitness function offloaded to the high performance processor.

Efficient partitioning of image registration system across the platforms and further optimisations to meet the real-time constraints of the application is future work. However, the concept of image registration system based on a partitioned genetic algoirthm has been proven in this work. This work can be taken and an integrated SOC using an ARM processor and a DSP produced.

6.9 Conclusions

In this Chapter an evolvable hardware system for detecting the orientation and displacement of Integrated Circuits has been presented. This image registration process is an adaptive process which uses a GA. The GA derives transforms for matching images of Integrated Circuits. The Chapter has presented the GA and the main stages designed to reduce the execution time of the algorithm. The Chapter goes on to give results for the execution of the algorithm on both the host system and a target SOC. The results show that an enhanced target platform is required to speed the execution of the fitness and hill climbing functions of the GA. An enhanced target platform is then described with software block function partitioning given. Analysis of the execution time of performance critical code sections shows that the use of a high performance DSP IP core will mean that the performance targets of the application can be met.

We have demonstrated here in this work the feasibility of running complex EHW tasks on a conventional SOC platform, hence making the need for specially tailored one-off customised hardware architectures unnecessary.

```
START MAIN BLOCK D /* main loop for genetic algorithm */
        Loop for 1 to number of generations or convergence for all ICs
                Loop for 1 to population size
                  select two individuals
                  apply crossover between rule sets
                  apply mutation

                  START SUB BLOCK E ***DSP*** /* calculate chromosome fitness */
                  Loop for 1 to number of ICs
                          get rotation from chromosome
                          index into rotation list for IC
                          get displacement from chromosome
                          displace IC image in rotation list
                          match captured to reference IC in rotation list
                          store chromosome fitness
                  End loop
                  END SUB BLOCK E

                End Loop

        START SUB BLOCK F /* Apply elitism */
                select top 10 % fittest chromosomes from old population
                to survive to new population
        END SUB BLOCK F

        START SUB BLOCK G ***DSP*** /* Apply hill climbing */
                select top 20 % fittest chromosomes
                For each chromosome loop for 1 to number of ICs
                        search hill climbing space modifying chromosome
                        calculate new fitness
                End loop
                if fitter chromosome found then re-encode chromosome
                        insert hill climbed individual into population
        END SUB BLOCK G

        End Loop
END MAIN BLOCK D

START BLOCK H /* Output results of search */
        Loop for 1 to number of ICs
          If no match found between reference and captured then IC
          rotation and displacement must be outside of tolerence - report error
          to operator
        End loop
END BLOCK H
```

Pseudo code 8: *Multiple IC Chromosome Encoding Algorithm Part 2*

```
/* the following are the orginal floating point sine and cosine */
/*dbCosTheta = cos( dbTheta );*/
/*dbSinTheta = sin( dbTheta );*/

/* i and j are integer representations of the sine and cosine */
i = 65536 * cos( dbTheta );
j = 65536 * sin( dbTheta );

loop nX = -(source-image.width / 2.0) and nPosX for 0
to source-image.width
        loop nY = -(source-image.height / 2.0) to
                source-image.height and nPosY = 0

                /* The following is the original floating point calculation */
                /*dbNewX = nX * dbCosTheta - nY * dbSinTheta + dbCentreX;*/
                /*dbNewY = nX * dbSinTheta + nY * dbCosTheta + dbCentreY;*/

                /* Integer Rotation applied to coordinates nX and nY (centered on dbCentreX
                and dbCentreY) */
                nNewX = ((nX * i - nY * j) >> 16) + dbCentreX
                nNewY = ((nX * j + nY * i) >> 16) + dbCentreY

                set intensity of result-image(nPosX , nPosY) with intensity from
                source-image(nNewX , nNewY)
                increment nX , nY and nPosX, nPosY
        end loop
end loop
```

Pseudo code 9: *Target rotation pseudo code*

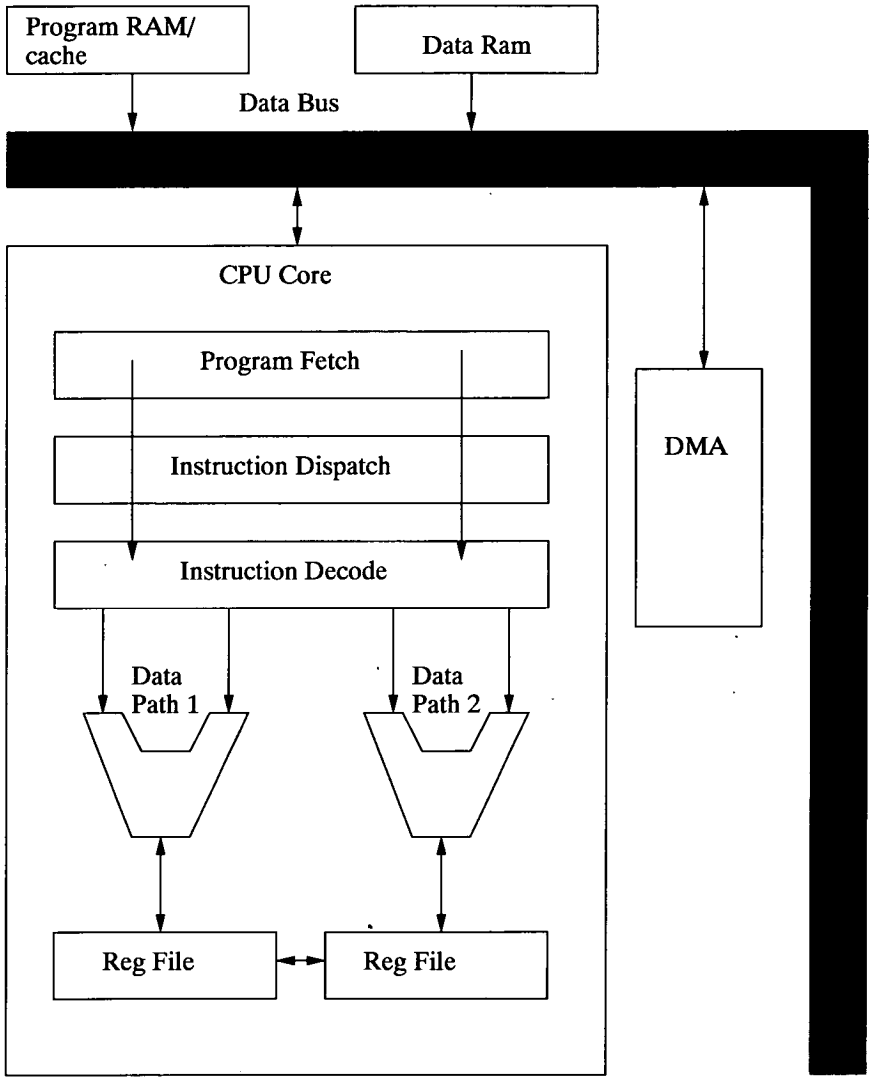

Figure 6.6: *DSP Architecture [3]*

Chapter 7
Summary and Conclusions

7.1 Introduction

This thesis has considered the use of an Evolvable Hardware system for Automatic Optical Inspection of Printed Circuit Boards. Embedded software running on programmable cores has been evaluated against the stringent real time constraints of the application. Novel image processing functions have been developed including heuristic techniques for object recognition and a GA for image registration.

The remainder of this chapter is organised as follows: in Section 7.2 summarises the thesis content and identifies the main contributions; in Section 7.3 the main achievements of this work are detailed; in Section 7.4 draws conclusions from the work presented in this thesis; in Section 7.5 discusses possible topics for future work and some final comments are made in Section 7.6.

7.2 Summary of Thesis

The theme of this thesis is the use of EHW, based on SOC technology and advanced image processing techniques, for a low cost AOI System for generating diagnostics for PCBs.

Chapter 2 discussed a number of techniques in Machine Vision and Evolutionary Computing relevant to this work. A central issue in the presentation of the techniques is the complexity of the algorithms. Efficiency is a major issue in this work because of the implications on the architecture and therefore cost of the embedded AOI system. The system also has stringent performance requirements with the processing and diagnostics of a complex test sample to be completed in one or two minutes.

Chapter 2 presented Machine Vision algorithms for edge detection, segmentation methods and image registration and discusses their complexity. Efficient edge detection methods are essential in this work for both the object recognition and image registration processes. Edge

detection operators find significant changes in intensity in a test sample image. These changes in intensity correspond to the edges of objects within an image, such as the edges of ICs.

The results of the edge detection phase are used as input into the higher level segmentation processes. Segmentation divides an image up into areas which correspond to objects in the scene. Robust segmentation techniques and their complexity are presented. The results of the segmentation techniques are then used in the higher level 2D object representation and recognition processes. Chapter 2 describes published processes and their relevance to this work.

The problem of image registration, which is central to an inspection system, is then presented. It is shown that traditional methods based on correlation become inefficient for the inspection problem in this work. Chapter 2 goes on to present evolutionary methods, specifically GAs, and ways of optimising them. A GA is used in this work for solving an image registration problem. The theoretical reasons for why GAs are successful for solving complex optimisation problems is given and this is based on schema theory. Existing work in the use of GAs for image registration is then detailed.

Chapter 2 gives a description of the field of EHW. The difference between gate level and functional level EHW is stated. How the EHW system presented in this work differs from the majority of the existing work in the field is given.

Chapter 2 concludes with a description of the existing work in AOI published in papers and in filed patents.

Chapter 3 discusses issues in real-time systems and SOC technology. The system in this work is a soft real-time system. The type of processing for the system is processing of a batch of test samples and is therefore sequential in nature.

The use of SOC technology has been selected for this project because of the performance benefits for running complex algorithms. These performance benefits take precedence over the other benefits of SOC technology for the AOI application, such as area and power consumption reduction.

The important area of SOC development methodologies is then discussed with the relevance to the AOI development and the issue of re-use. Re-use is a central issue in SOC development due to increased time scale pressures and product functionality requirements. The HASoC methodology [2] is discussed in detail as it raises many issues applicable to the AOI work. The

method by which HASoC supports Re-use is highlighted.

Chapter 3 concludes with a discussion of SOC technology for vision systems. It is emphasised that the use of SOC technology executing complex machine vision algorithms for an AOI system is novel work.

Chapter 4 presents a detailed analysis of the use of Hough Transforms for object recognition, applied specifically to the problem of detecting Integrated Circuits within complex images of Printed Circuit Boards. Two techniques are developed and analysed for their computational resource requirements and accuracy. The first technique is the original Hough Transform for detecting rectangles which correspond to the edges of the Integrated Circuits. With the Hough Transform technique the image is scanned and for each pixel there is a vote into accumulator space. The number of potential ICs for each pixel point is very large and the accumulator array is four dimensional. The complexity of this algorithm was found to be $O(n^6)$. The algorithm is therefore too complex for complex images both in terms of processing time and memory resource.

A second reduced complexity technique was developed which was still based on a Hough Transform but for detecting lines rather than rectangles. Having detected lines in an image a technique based on Gaussian Elimination was used for detecting cross point between lines. From these cross points rectangles are detected which correspond to edges of ICs. The reduced complexity technique was found to be neither accurate or fast enough for complex images. This problem was due to the edges of the chips in the edge map not being clearly defined and therefore they register as multiple lines. A further problem was due to the Chip ID in the centre of the chips also registering as lines. Chapter 4 ends with the use of noise reduction techniques and morphological operators for removing the pixels which cause the technique to fail. This approach only met with limited success, with important edge information being removed as well as the Chip ID.

In Chapter 5 reduced complexity segmentation techniques based on Region Growing are developed. A key feature of these algorithms is they produce accurate results without the assistance of CAD data. The algorithms are targeted for an ARM SOC platform. The algorithms are found to be of complexity $O(n)$ which represents a significant reduction in complexity compared to the Hough Transform based techniques. This reduction in complexity makes them suitable for use in an embedded application.

The region growing technique to find Chip positions works by first generating a binary image through band thresholding the original grey scale image. The technique then proceeds to grow regions in areas of zero intensity. This region map is then processed through heuristics that Chips ID regions are closely grouped together. All other regions are deleted to give a Chip ID map. From the start points of the remaining regions approximate Chip Centroid positions are calculated. The Chip ID map is then applied as a mask to the edge map to delete the Chip ID. A final stage is to search from the computed approximate chip centroid positions in the edge map to find the edges of the Chips. This technique has been shown to work in complex test samples which contain many other features other than Chip ID.

The major fault with the region growing process is that it only detects ICs within a restricted band of intensities. Further pre-processing operators are presented in Chapter 5 to detect ICs made of varying materials, dark grey plastic or metallic for instance. The first technique involves using the result of the first application of the SUSAN edge detection operator to segment the original grey scale image. This technique is efficient and produces improved results for IC detection.

The second technique is more complex and involves probability processing. This technique starts with the derivation of the Cumulative Probability Function (CPF) of an image and then transforming the intensity distribution of the image so that every pixel is proportional to its CPF. This isolates the Chips from the background thus supporting further processing. A Region of Interest is then selected which corresponds to a Chip. A Gaussian probability function is then derived from this ROI. This probability function is then applied to every pixel in the transformed image. This generates an image whose intensity is directly proportional to the probability that it belongs to the distribution. Through thresholding this final image a binary map is generated which can then be processed by the Region Growing Technique as before.

As well as object recognition an important class of algorithms for an inspection system are image registration algorithms. These algorithms can be used to check for highly accurate placement of components on PCBs. Chapter 6 discusses techniques for checking the correct orientation and displacement of Integrated Circuits using a Genetic Algorithm. As with the object recognition work the efficiency of the algorithms is a major issue. Major stages in the optimisation of the algorithm are discussed which include restricting the search space for the algorithm. The search space can be restricted through knowledge of the application area. ICs are placed within a very fine tolerance for them to operate correctly therefore the length of

the chromosomes processed by the Genetic Algorithm can be restricted. Through a restricted search space the algorithm converges faster.

The image registration algorithms are targeted for an Evolvable Hardware platform based on an ARM7 SOC. To meet the performance requirements of the application the use of a high performance DSP IP core to enhance the SOC processor is proposed. Execution studies of the Genetic Algorithm on a Texas EVM6201 DSP are carried out. It is shown that through using an enhancement it can be predicted that the performance requirements of the application can be met. Through the software partitioning across the enhanced platform the DSP executes the computationally expensive parts of the algorithm. These processor intensive tasks are the pixel match procedure in the GA's fitness function.

7.3 Summary of Achievements

The main achievement of this work is the development of novel object recognition and image registration algorithms and their optimisation and targeting to embedded architectures, based on SOC technology. Image processing functions are demanding for even the most advanced embedded platform. A major achievement of the work is to predict that through optimisation of the software and the use of SOC technology, with an advanced DSP enhancement, the performance criteria of a PCB inspection system can be met.

The following is a breakdown of work packages developed for this thesis. The total number of lines of C and C++ code in each work package are given. The number of lines given does not include the code from the image processing library, described in [21], as this code was used without changes.

7.3.1 Host Implementations

7.3.1.1 Novel reduced complexity Hough Transform technique.

This technique is based around detecting lines in an edge map. By finding the cross points between lines, rectangles can be detected which correspond to the edges of Integrated Circuits. This technique has significantly reduced complexity, but is still not efficient enough for the demands of the application and does not produce accurate enough results.

7.3.1.2 Novel heuristic technique based on region growing.

This technique uses region growing which is a low complexity technique. Through finding regions which correspond to Chip IDs the ICs can be found. The technique is efficient and the results are robust and accurate for complex test samples giving effective recognition of ICs.

7.3.1.3 Novel optimised Genetic Algorithm for Image Registration.

This algorithm detects rotation and displacement errors of ICs. The GA is used to derive transformations which map a captured image to a reference image. The algorithm is optimised and the execution times for accurate diagnostics of complex images are within bounds for the demands of the application. A significant proportion of the code for this implementation was taken from a GA library.

7.3.1.4 Novel enhanced segmentation technique.

This is a low complexity technique for detecting ICs within widely varying bands of intensity. The technique uses the output of the SUSAN edge detection operator to segment the original image. The results give enhanced accuracy of object recognition without significant computational overhead.

7.3.2 Target implementations

The code from the image processing library used by the object recognition and image registration code was ported to the target development environments. These development environments are ADS for the SOC platform and CCS for the DSP.

The following work packages are specific to the target implementation.

7.3.2.1 Optimised Genetic Algorithm for Image Registration targeted for an SOC platform.

No further major code changes for this implementation.

7.3.2.2 Optimised Genetic Algorithm for Image Registration targeted for a Digital Signal Processor.

No further major code changes for this implementation.

7.3.2.3 Novel genetic algorithm for Image Registration partitioned across an enhanced SOC platform.

The additional code for this implementation is the file handling code for enabling communication between the SOC platform and the DSP platform.

7.4 Conclusions

This thesis proposed that an Evolvable Hardware System can be built for a low cost Automatic Optical Inspection system out of standard programmable cores and advanced image processing techniques in software.

From the work in Chapter 4 this thesis concludes that the use of the original Hough Transform segmentation technique for the application presented in this work is too computationally complex. The complexity of the algorithms is found to be of order O(n^6). Chapter 4 goes on to develop a reduced complexity technique still based on using the Hough Transform for detecting lines and then rectangles which correspond to the borders of ICs. This technique was neither found to be accurate or fast enough for the AOI application. Chapter 4 ends with the use of noise reduction and morphological operators but it was found they did not assist the object recognition process.

From the work in Chapter 5 this thesis concludes that a segmentation technique based on Region Growing produces accurate and fast recognition of ICs within complex test samples. The Region Growing algorithms have complexity O(n) compared to order O(n^6) for the Hough Transform. This reduction in complexity allows the use of such algorithms in embedded systems which is a major goal of this work. A key feature of the Region Growing algorithms is that they work without the assistance of CAD data. This thesis therefore concludes that object recognition within complex images of PCBs is possible without the assistance of CAD data.

Chapter 5 has presented the porting of the techniques to an ARM7 SOC target. The porting of

an object recognition operator to this target system is novel and the feasibility of this approach has been shown.

Chapter 5 also shows the limitations of the Region Growing technique and two major enhancements are presented. The first is based on segmentation of an image through using the results of the SUSAN edge detection operator. The second is a higher complexity technique based on probability processing. Chapter 5 also gives the conclusion that these techniques can assist greatly in the object recognition process.

Chapter 6 presents an EHW system for detecting the orientation and displacement of ICs. Optimisation stages for the software and performance studies are presented. A conclusion is an enhancement to the SOC platform is required consisting of a high performance DSP IP core for the performance requirements of the application to be met.

Chapter 6 demonstrates the feasibility of running complex EHW tasks on a conventional SOC platform, hence making the need for specially tailored one-off customised hardware architectures unnecessary. This is a key result and conclusion for this thesis.

7.5 Future Work

Having proved the concept of a low cost Automatic Optical Inspection system, a System-On-Chip platform integrating a micro-controller and a high performance Digital Signal Processor can be developed. This will give significant performance improvements over the existing platform. The software will have to be optimised onto the new platform and a RTOS integrated to manage the software components and the interface to the user. The communication between the SOC and the on-chip DSP will have to be defined. There are alternatives for selection of processor for synthesis, including ARM cores [87] and the Leon processor [118].

The object recognition phase should be enhanced to include resistors, capacitors, connectors and other typical features on a board. The image registration system could also perform checking of displacement and rotation errors of these further components. Other types of error on PCBs including raised components could be detected, however this would require a multi-camera system for detection with 3D image processing functions.

The image registration procedure can be optimised through performing registration just on the corners of the ICs. The object recognition phase finds the positions of the corners of the ICs

so the registration procedure can just be performed on the corners of the chips. Therefore, the number of pixel transformations can be greatly reduced.

The computationally expensive parts of the image registration process , such as the fitness function, could be hand coded in assembly language. This kind of optimisation is common for embedded SOC applications. However, this assembly code would be tied to a particular family of processors, where as all the existing library code can be easily targeted for a variety of processors.

The existing hardware platform has the ARM Integrator platform and the Texas EVM6201 DSP board communicating via the file system on the host PC. In this communication parameters are passed during the GA processing. Communication via the file system is sub-optimal and direct connection between the two platforms should be investigated.

For the AOI system to be used on production lines the system needs a Graphical User Interface [12]. This interface needs to take into account the fact that the operator of the AOI machine may not be from a programming background. Example user input would include the correction of the object recognition phase if the processing is not 100% accurate. Further interface functions would include triggering of the image capture and monitoring the progress of the processing of the test samples.

The current system implementation assumes the operator aligns the test samples precisely on a grid before image capture, therefore an initial alignment and registration procedure as described in [20] is not necessary. However, on a high throughput production line the samples will pass on a conveyor belt before image capture. In this case an initial alignment step will be necessary before the registration phase otherwise accurate diagnostics will not be possible.

7.6 Final Remarks

In this work the concept of a low cost Automatic Optical Inspection system based on Evolvable Hardware has been proven. A library of image processing functions optimised for embedded processors has been developed. The functions have been proven to give accurate diagnostics against multiple test samples.

The work forms the basis of a high functionality product for giving accurate diagnostics of Printed Circuit Boards within the real-time constraints of the application.

References

[1] R. Gonzalez and R. Woods, *"Digital Image Processing"*. Addison-Wesley, 1993.

[2] P. Green, "HASoC - Towards a New Method for System-On-a-Chip Development," in *Design Automation for Embedded Systems*, vol. 6, pp. 333 – 353, July 2002.

[3] "Texas Instruments." http://dspvillage.ti.com - As viewed on 26/7/04.

[4] R. Garnick and I. Syed, "Optical Test for Optimum Quality." Electronics Manufacture and Test, November 2000.

[5] D. Haigh, "Repeatability is AOI Watchword for Volume SMT Production." Test and Measurement Europe, December/January 2000.

[6] J. Arena, "Investing in Inspection." Test, December/January 2000.

[7] I. Sommerville, *"Software Engineering"*. Addison-Wesley, 6th edition ed., 2001.

[8] A. Shaw, *"Real-Time Systems and Software"*. John Wiley and Sons, 2001.

[9] B. P. Douglass, *"Doing Hard Time - Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns"*. Addison-Wesley, 1999.

[10] A. Burns, *"Real-Time Systems and Programming Languages : Ada 95, Real-Time Java, and Real-Time POSIX"*. Addison-Wesley, 3rd ed., 2001.

[11] R. Frank, M. Wood, and R. Barrett, "Introduction to Real Time Systems." Hatfield Polytechnic: Division of Computer Science, 1991. 2nd Edition.

[12] A. Dix, J. Finlay, G. Abowd, and R. Beale, *"Human-Computer Interaction"*. Prentice Hall, 1998.

[13] Contax Ltd. http://www.contax.co.uk - As viewed on 26/704.

[14] Teradyne Inc. http://www.teradyne.com - As viewed on 26/7/04.

[15] Camtek Ltd. http://www.camtek.co.il - As viewed on 26/7/04.

[16] Qualelectron Systems Corporation. http://www.qualectron.com/AOIHome.htm - As viewed on 26/7/04.

[17] Diagnosys Ltd. http://www.diagnosys.com - As viewed on 26/7/04.

[18] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd, *"Surviving the SOC Revolution: A Guide to Platform-Based Design"*. Kluwer Academic Publishers, 1999.

[19] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, N. Salami, N. Kajihara, and N. Otsu, "Real-World Applications of Analog and Digital Evolvable Hardware," *IEEE Transactions on Evolutionary Computation*, vol. 3, September 1999.

[20] L. Brown, "A Survey of Image Registration Techniques," *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, 1992.

[21] A. Peacock, *"Information Fusion for Improved Motion Estimation"*. PhD thesis, School of Engineering and Electronics, University of Edinburgh, May 2001.

[22] B. Kernighan and D. Ritchie, *"The C Programming Language"*. Prentice Hall, 1988.

[23] B. Stroustrup, *"C++ Programming Language"*. Addison Wesley, 3rd ed., 1997.

[24] Zot Engineering Ltd. http://www.zot.co.uk - As viewed on 26/7/04.

[25] M. Sonka, V. Hlavac, and R. Boyle, *"Image Processing, Analysis, and Machine Vision"*. PWS Publishing, 2nd ed., 1999.

[26] J. Evans and T. Arslan, "Enhanced Image Detection on an ARM Based Embedded System," *Design Automation for Embedded Systems: Special Issue on Embedded System Design in the UK, Kluwer Academic Publishers*, vol. 6, pp. 477–487, July 2002.

[27] D. Ballard and C. Brown, *"Computer Vision"*. Prentice-Hall, 1982.

[28] T. Back, D. Fogel, and T. Michalewicz, eds., *"Evolutionary Computation 1"*. Institute of Physics Publishing, 2000.

[29] T. Back, D. Fogel, and T. Michalewicz, eds., *"Evolutionary Computation 2"*. Institute of Physics Publishing, 2000.

[30] J. Holland, *"Adaptation in Natural and Artificial Systems"*. The MIT Press, 1992.

[31] M. Mitchell, *"An Introduction to Genetic Algorithms"*. The MIT Press, 2002.

[32] D. Goldberg, *"Genetic Algorithms"*. Addison-Wesley publishing company, 1989.

[33] S.Smith and J.Brady, "SUSAN - A New Approach to Low Level Image Processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[34] J.F.Canny, "A Computational Approach to Edge Detection," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[35] S. Bhandarkar and H. Zhang, "Image Segmentation Using Evolutionary Computation," *IEEE Transactions on Evolutionary Computation*, vol. 3, April 1999.

[36] M. Brady and H. Asada, "Smoothed Local Symmetries and their Implementation, AI Memo 757," tech. rep., Massachusettes Institute of Technology, Artificial Intelligence Laboratory, February 1984.

[37] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, January 1986.

[38] F. Stein and G. Medioni, "Structural Indexing: Efficient 2D Object Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, December 1992.

[39] M. Marefat and R. Kashyap, "Image interpretation and object recognition in manufacturing," *IEEE Control Systems Magazine*, vol. 11, pp. 8–17, August 1991.

[40] U. Erdem, H. Civi, and A. Ercil, "2D Object Recognition using Implicit Polynomials and Algebraic Invariants," in *9th Mediterranean Electrotechnical Conference, MELECON 98*, vol. 1, pp. 53–57, May 1998.

[41] J. Evans and T. Arslan, "The Implementation of an Evolvable Hardware System for Real Time Image Registration on a System-On-Chip Platform," in *NASA/DOD Conference on Evolvable Hardware (EH2002)*, pp. 142–146, 2002.

[42] J. Evans and T. Arslan, "Implementation of a Robust Image Registration Algorithm on an ARM System-On-Chip platform," in *IEEE2002 International Symposium on Cricuits and Systems (ISCAS2002)*, vol. 2, pp. 269–272, 2002.

[43] G. Pagliari and J. Greene, "Image Registration, Parameter Tuning and Approximate Function Evaluation, using the Genetic Algorithm and Digital Image Warping," in *IEEE AFRICON 4th*, vol. 2, pp. 536–541, September 1996.

[44] D. Robinson, "Correction of Illumination Variations in Template Type Operations." Integral Vision, Discussion Document, September 2000.

[45] C. K. Chow, H. T. Tsui, T. Lee, and T. K. Lau, "Medical Image Registration and Model Construction using Genetic Algorithms," in *Proceedings of the International Workshop on Medical Imaging and Augmented Reality*, pp. 174–179, 2001.

[46] A. Mahajan, A. Pilch, and T. Chu, "Intelligent Image Correlation using Genetic Algorithms for Measuring Surface Deformations in the Autonomous Inspection of Structures," in *American Control Conference*, vol. 1, pp. 460–461, September 2000.

[47] J. Grefenstette, J. Fitzpatrick, and D. van Gucht, "Image Registration by Genetic Search," in *Proceedings of IEEE Southeast Conference*, pp. 460–464, 1984.

[48] P. Chalermwat and T. El-Ghazawi, "Multi-resolution Image Registration Using Genetics," in *Proceedings of the International Conference on Image Processing ICIP 1999*, pp. 452–456, 1999.

[49] H. Ammar and Y. Tao, "Fingerprint Registration using Genetic Algorithms," in *Proceedings 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, 2000*, pp. 148–154, 2000.

[50] J. Grefensttete and J. Fitzpatrick, "Genetic Search with Approximate Function Evaluations," in *Proceedings of an International Conference on Genetic Algorithms and their Applications, 1985*, pp. 112–120, 1985.

[51] J. Marsden and A.Tromba, *"Vector Calculus"*. W.H. Freeman and Company, 5th ed., 2003.

[52] B. Turton, T. Arslan, and D. Horrocks, "A Hardware Architecture for a Parallel Genetic Algorithm for Image Registration," in *IEE Colloquium on Genetic Algorithms in Image Processing and Vision*, pp. 11/1 – 11/6, 1994.

[53] B. Manderick and T. Higuchi, "Evolvable Hardware: An Outlook," in *Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science 1259*, pp. 305–310, Springer-Verlag, 1997.

[54] M. Murakawa, S. Yoshizawa, I. Kajitani, X. Yao, N. Kajihara, M. Iwata, and T. Higuchi, "The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing," *IEEE Transactions on Computers*, vol. 48, no. 6, 1999.

[55] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi, "Hardware Evolution at Function Level," in *International Conference on Evolutionary Computation. 4th Conference on Parallel Problem Solving from Nature - PPSN IV , LNCS 1141*, pp. 62–71, September 1996.

[56] B. Hounsell, *"Programmable Architectures for the Automated Design of Digital FIR Filters using Evolvable Hardware"*. PhD thesis, The University of Edinburgh, September 2001.

[57] M. Salami, M. Murakawa, and T. Higuchi, "Lossy Image Compression by Evolvable Hardware," in *proceedings of IJCAI-97 Workshop on Evolvable Systems*, pp. 53–59, 1997.

[58] H. Sakanashi, M. Salami, M. Iwata, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, and T. Higuchi, "Evolvable Hardware Chip for High Precision Printer Image Compression," in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[59] M. Salami, H. Sakanashi, M. Iwata, T. Kurita, and T. Higuchi, "On-line Compression of High Precision Printer Images by Evolvable Hardware," in *Proceedings of Data Compression Conference*, pp. 219–228, 1998.

[60] W. Wolf, *"Computers as Components: Principles of Embedded Computing System Design"*. Morgan Kaufmann Publishers, 2001.

[61] J. Hennessy and D. Patterson, *"Computer Architecture A Quantitative Approach"*. Morgan Kaufmann Publishers, Inc., 2nd ed., 1996.

[62] W. Qin, S. Rajagopalan, M. Vachharajani, H. Wang, X. Zhu, D. August, K. Keutzer, S. Malik, and L.-S. Peh, "Design Tools for Application Specific Embedded Processors," in *Second International Conference on Embedded Software, EMSOFT2002*, pp. 319–333, October 2002.

[63] H. Loh and M. Lu, "Printed Circuit Board inspection using image analysis," in *International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies, 1995*, pp. 673–677, May 1995.

[64] M. Zervakis, S. Goumas, and G. Rovithakis, "A bayesian framework for multilead smd post-placement quality inspection," *Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics : Accepted for future publication*, 2003.

[65] A. Hayter, *"Probability and Statistics: for Engineers and Scientists"*. Duxbury, Thomson Learning, Inc, 2nd ed., 2002.

[66] N.Kim, J.Pyun, K.Choi, B.Choi, and S.Ko, "Real-time Inspection System for Printed Circuit Boards," in *IEEE International Symposium on Industrial Electronics. ISIE 2001*, vol. 1, pp. 166–170, June 2001.

[67] R. Straayer, D. Seniff, P. Walsh, J. Gerber, J. Kohler, S. Snietka, and B. Davidson, "Automatic Optical Inspection System having a Weighted Transition Database." US Patent Office, Patent Number: US5608453, Assignee: Gerber Systems Corp, March 1997.

[68] Forgues, M. Pierre, Prasada, and Birendra, "Automatic Optical Inspection System." US Patent Office, Patent Number: US4794647, Assignee: Northern Telecom Limited (Montreal, CA), December 1988.

[69] T. Yotsuya and H.Takahara, "Input Method for Reference Printed Circuit Board Assembly Data to an Image Processing Printed Circuit Board Assembly Automatic Inspection Apparatus." US Patent Office, Patent Number: US 4894790, Assignee: Omron Tateisi Electronics Co. (Kyoto, JP), January 1990.

[70] H. Nishikawa, "Printed Circuit Board Inspection Apparatus and Method." US Patent Office, Patent Number: US 6151063, Assignee: Nidec Read Corporation (Kyoto, JP), November 2000.

[71] P. Seng, "Method and an Apparatus for Inspection of a Printed Circuit Board Assembly." US Patent Office, Patent Number: US 6084663, Assignee: Hewlett-Packard Company (Palo Alto, CA), July 2000.

[72] H. Wasserman, "Apparatus and Method for Illuminating a Printed Circuit Board for Inspection." US Patent Office, Patent Number: US 5060065, Assignee: Cimflex Teknowledge Corporation (Princeton, NJ), October 1991.

[73] D. Sepai, K. Daly, B. Whalen, K. Hong, and G. Jones, "Apparatus and Method for Inspection of High Component Density Printed Circuit Board." US Patent Office, Patent Number: US 5455870, Assignee: Raytheon Company (Lexington, MA), October 1995.

[74] J. Lawrence, W. Persons, G. Preckshot, and J. Gallagher, "Evaluating Software for Safety Systems in Nuclear Power Plants," in *Proceedings of the Ninth Annual Conference on Computer Assurance, COMPASS '94 'Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security'*, pp. 197–207, June-July 1994.

[75] Cadence Design Systems, Inc. http://www.cadence.com - As viewed on 26/7/04.

[76] M. Verderber, A. Zemva, and A. Trost, "HW/SW Codesign of the MPEG-2 Video Decoder," in *Proceedings of the International Symposium on Parallel and Distributed Processing*, pp. 179–185, April 2003.

[77] D. Smith, *"HDL Chip Design: A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs using VHDL or Verilog"*. Doone Publications, 1996.

[78] G. Martin and C. Lennard, "Improving Embedded Software Design and Integration in SOCs," in *Proceedings of the IEEE Custom Integrated Circuits Conference, CICC*, pp. 101–108, May 2000.

[79] P. Stevens and R. Pooley, *"Using UML: Software Engineering with Objects and Components"*. Addison-Wesley, updated edition ed., 2000.

[80] S. Furber, *"ARM System Architecture"*. addison-wesley, 1996.

[81] www.ti.com - As viewed on 26/7/04, *"TMS320C6000 Technical Brief, SPRU197D"*, February 1999.

[82] www.ti.com - As viewed on 26/7/04, *"TMS320C6000 Programmers Guide, SPRU198F"*, October 2000.

[83] www.ti.com - As viewed on 26/7/04, *"TMS320C6000 CPU and Instruction Set Reference Guide, SPRU189F"*, October 2000.

[84] V. Zivkovic and P. Lieverse, "An Overview of Methodologies and Tools in the Field of System-Level Design," in *Embedded Processor Design Challenges: Systems, Architectures, Modeling and Simulation SAMOS 2001*, vol. LNCS 2268, pp. 74–88, 2002.

[85] ARM Ltd, *"ARM Integrator/AP: ASIC Development Motherboard User Guide, ARM DUI 0098 A"*.

[86] ARM Ltd, *"ARM Integrator/CM7TDMI User Guide, ARM DUI 0126 A"*.

[87] ARM Ltd. http://www.arm.com - As viewed on 26/7/04.

[88] A. Sharma and C. Ravikumar, "Efficient Implementation of ADPCM Codec," in *Thirteenth International Conference on VLSI Design*, pp. 456–461, January 2000.

[89] G. Booch, *"Object Oriented Analysis and Design with Applications"*. Benjamin Cummings, 1994.

[90] W. Fang, "A System-On-Chip Design of a Low-Power Smart Vision System," in *IEEE Workshop on Signal Processing Systems, SIPS 98*, pp. 63–72, October 1998.

[91] J. Skribanowitz, T. Knobloch, J. Schreiter, and A. Konig, "VLSI Implementation of an Application-Specific Vision Chip for Overtake Monitoring, Real Time Eye Tracking, and Automated Visual Inspection," in *Proceedings of the Seventh International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems, MicroNeuro '99.*, pp. 45–52, April 1999.

[92] Mobil Eye N.V. http://www.mobileye.com - As viewed on 26/7/04.

[93] C. Torres-Huitzil, S. Maya-Rueda, and M. Arias-Estrada, "A Reconfigurable Vision System for Real-time Applications," in *Proceedings 2002 IEEE International Conference on Field-programmable Technology*, pp. 286–289, 2002.

[94] R. Duda and P. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures," *Commun. ACM*, vol. 15, pp. 11–15, 1972.

[95] B. Gatos, S. Perantonis, and N. Papamarkos, "Accelerated Hough Transform Using Rectangular Image Decomposition," *Electronic Letters*, vol. 32, April 1996.

[96] N. Guil, J.Villalba, and E. Zapata, "A Fast Hough Transform for Segment Detection," *IEEE Trans. Image Process*, vol. 4, no. 11, pp. 1541–1548, 1995.

[97] J.E.Vuillemin, "Fast Linear Hough Transform," in *Proceedings of the International Conference on Application Specific Array Processors*, pp. 1–9, 1994.

[98] D. Robinson, "Fast Hough Transform in Greyscale Images." Integral Vision, Discussion Document, October 1996.

[99] K.Stroud, *"Engineering Mathematics"*. Macmillan Press Ltd, 4th ed., 1995.

[100] H.Loh and M.Lu, "Printed Circuit Board Inspection Using Image Analysis," *IEEE Transactions on Industry Applications*, vol. 35, no. 2, 1999.

[101] C.Kannan and H.Chuang, "Fast Hough Transform on a Mesh Connected Processor Array," *Inf. Proc. Lett*, vol. 33, no. 5, pp. 243–248, 1990.

[102] L. Ke-qinq and T. Qi, "A Fast Hough Transform for Inspecting Accurate Needle-type Meter Guages," in *Proc. IAPR Workshop on Computer Vision*, pp. 195–198, 1988.

[103] R.M.Haralick and L. Shapiro, "Image Segmentation Techniques," *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 100–132, 1985.

[104] ARM Ltd, *"Multi-ICE User Guide, ARM DUI 0048D"*, 2 ed.

[105] ARM Ltd, *"ARM Developer Suite, Assembler Guide, ARM DUI 0068A"*, 1.1 ed., 2000.

[106] ARM Ltd, *"ARM Developer Suite: Compiler, Linker and Utilities Guide, ARM DUI 0067C"*, 1.1 ed.

[107] ARM Ltd, *"ARM Developer Suite: Debug Target Guide, ARM DUI0058C"*, 1.1 ed.

[108] ARM Ltd., *"ARM Developer Suite: Developer Guide, ARM DUI 0056C"*, 1.1 ed.

[109] ARM Ltd., *"ARM Developer Suite: Codewarrior IDE Guide, ARM DUI 0065C"*, 1.1 ed.

[110] ARM Ltd., *"ARM Developer Suite: Debuggers Guide, ARM DUI 0066C"*, 1.1 ed.

[111] "XV." Anonymous FTP from ftp.cis.upenn.edu directory pub/xv - Correct as of 26/7/04.

[112] "Gimp." http://www.gimp.org - As viewed on 26/7/04.

[113] D. Robinson, "Probability Processing." Integral Vision, Active Document, February 2001.

[114] A. Mehrotra, R. Srikanth, and A. Ramakrishnan, "A New Coding Scheme for 2-d and 3-d mr Images using Shape Adaptive Integer Wavelet Transform," in *Proceedings of International Conference on Intelligent Sensing and Information Processing*, pp. 67–72, Jan 2004.

[115] L. Tao and H. Kwan, "Real-valued Discrete Gabor Transform for Image Representation," in *The 2001 IEEE International Symposium on Circuits and Systems (ISCAS 2001)*, vol. 2, pp. 589–592, May 2001.

[116] T. Asakura, Y. Aoyagi, and O. Hirose, "Real-Time Recognition of Road Traffic Sign in Moving Scene Image using new Image Filter," in *SICE 2000. Proceedings of the 39th SICE Annual Conference. International Session Papers*, 2000.

[117] Berkeley Design Technology Inc, http://www.bdti.com/index.html - As viewed on 26/7/04.

[118] Gaisler Research. http://www.gaisler.com - As viewed on 26/7/04.